# Chapter 6
# Multichannel Fast QRD-RLS Algorithms

António L. L. Ramos and Stefan Werner

**Abstract** When considering multichannel adaptive implementations, it is often possible to directly apply standard single-channel algorithms to the multichannel problem, e.g., the numerically stable and fast converging QR decomposition recursive least-square (QRD-RLS) algorithm. Even though such a solution would provide fast convergence, it may be computationally too complex due to a large number of coefficients. In order to obtain a computationally efficient solution, RLS-type algorithms specially tailored for the multichannel setup are a good option. This chapter introduces various multichannel fast QRD-RLS (MC-FQRD-RLS) algorithms that can be seen as extensions of the basic single-channel FQRD-RLS algorithms to the case of a multichannel input vector, where it can be assumed that each channel has a time-shift structure. We provide, in a general framework, a comprehensive and up-to-date discussion of the MC-FQRD-RLS algorithms, addressing issues such as derivation, implementation, and comparison in terms of computational complexity.

## 6.1 Introduction

Multichannel signal processing can be found in various applications such as color image processing, multispectral remote sensing imagery, biomedicine, channel equalization, stereophonic echo cancelation, multidimensional signal processing, Volterra-type non-linear system identification, and speech enhancement [1, 2]. When choosing among the adaptive algorithms that can cope with multichannel signals, the choice is more than often based on stability, convergence speed, and

António L. L. Ramos
Buskerud University College, Kongsberg – Norway
e-mail: antonio.ramos@hibu.no

Stefan Werner
Helsinki University of Technology, Helsinki – Finland
e-mail: stefan.werner@tkk.fi

J.A. Apolinário Jr. (ed.), *QRD-RLS Adaptive Filtering*,
DOI 10.1007/978-0-387-09734-3_6, 

computational complexity. The standard QR decomposition recursive least-squares (QRD-RLS) algorithm stands out as potential good option because of its well-known fast convergence property and excellent numerical behavior. However, its $\mathcal{O}[P^2]$ computational complexity makes its use prohibitive when higher order filters are required.

The FQRD-RLS algorithms, in general, offer the same fast converging feature as the standard QRD-RLS algorithm, while attaining a lower computational complexity, which is achieved by exploiting the underlying time-shift structure of the input signal vector. Historically, the first solutions were presented for the case where the input signal is just a "tapped-delay" line, i.e., a single-channel signal, and multichannel fast QRD-RLS (MC-FQRD-RLS) algorithms arise as a natural extensions of basic FQRD-RLS algorithms making them useful also in multichannel applications. The MC-FQRD-RLS algorithms can be classified into three distinct ways, according to [3]: (1) which error vector is being updated (*a priori* or *a posteriori*); (2) the type of prediction used (forward or backward),[1] and; (3) the approach taken for the derivation (sequential- or block-type). The first two are inherited from the single-channel case, whereas the last one, specific for multichannel algorithms, determines how new multichannel input samples are processed. These three concepts are combined in Table 6.1 for the case of MC-FQRD-RLS algorithms based on backward prediction errors, which are the ones addressed in this work. The structure parameter introduced in this table simply denotes the way a particular algorithm is implemented.

**Table 6.1** Classification of the MCFQRD-RLS algorithms.

| Error type | Approach and order | | Structure | References | Algorithm |
|---|---|---|---|---|---|
| **MCFQR POS_B** | **BLOCK-CHANNEL** | *Equal* | Lattice | [4] | **1** |
| | | *Order* | Transversal | [4–6] | **2** |
| | | *Multiple* | Lattice | — | **3** |
| | | *Order* | Transversal | [7, 8] | **4** |
| | **SEQUENTIAL-CHANNEL** | *Equal* | Lattice | Implicit in [9] | **5** |
| | | *Order* | Transversal | Suggested in [5] | **6** |
| | | *Multiple* | Lattice | [9] | **7** |
| | | *Order* | Transversal | [8, 10] | **8** |
| **MCFQR PRI_B** | **BLOCK-CHANNEL** | *Equal* | Lattice | [11] | **9** |
| | | *Order* | Transversal | [4, 6, 11] | **10** |
| | | *Multiple* | Lattice | — | **11** |
| | | *Order* | Transversal | [7] | **12** |
| | **SEQUENTIAL-CHANNEL** | *Equal* | Lattice | Implicit in [11] | **13** |
| | | *Order* | Transversal | Implicit in [11] | **14** |
| | | *Multiple* | Lattice | [11] | **15** |
| | | *Order* | Transversal | [11] | **16** |

[1] This chapter does not consider FQRD-RLS algorithms based on the updating of the forward error vector. As pointed out in Chapter 4, and still holding for the multichannel case, these algorithms are reported to be unstable, in contrast with their backward error vector updating-based counterparts.

Depending on the approach taken for the derivation, the $\mathcal{O}[P^2]$ computational complexity of the standard QRD-RLS implementation can be reduced to $\mathcal{O}[MP]$ and $\mathcal{O}[M^2P]$, for sequential- and block-channel processing algorithms, respectively; with $P$ being the total number of filter coefficients and $M$ the number of channels. Although the computational cost of block-channel algorithms is slightly higher than sequential-channel algorithms, they are more suitable for parallel processing implementations.

After reformulating the problem for the multichannel case, most of the equations are turned into matrix form, with some involving matrix inversion operations. Beside being potential sources of numerical instability, the computational burden is also greatly increased contrasting with desirable low-complexity feature of single-channel FQRD-RLS algorithms that rely on scalar operations only. Nevertheless, many good solutions to these problems exist and are addressed in the following.

The reminder of this chapter is organized as follows. In Section 6.2, we introduce the standard MC-FQRD-RLS problem formulation along with a new definition of the input vector that allows for a more general setup where the various channels may have different orders. The sequential- and block-type algorithms are discussed in Sections 6.3 and 6.4, respectively, while order-recursive implementations are addressed in Section 6.5. An application example and computational complexity issues are presented in Sections 6.6.1 and 6.6.2, respectively. Finally, closing remarks are given in Section 6.7.

## 6.2 Problem Formulation

The MC-FQRD-RLS problem for channels of equal orders was addressed in early 90's, in [5], where the sequential approach was introduced to avoid complicated matrix operations, and in [12] where both sequential and block approaches were addressed. Later, multiple-order block-multichannel algorithms comprising scalar operations only were introduced in [3, 8, 11, 13].

For the multichannel problem, the weighted least-squares (WLS) objective function, introduced in Chapter 2, is given as

$$\xi(k) = \sum_{i=0}^{k} \lambda^{k-i}[d(i) - \mathbf{x}_P^{\mathrm{T}}(i)\mathbf{w}_P(k)]^2 = \mathbf{e}^{\mathrm{T}}(k)\mathbf{e}(k), \tag{6.1}$$

where $\mathbf{e}(k)$ is the weighted error vector defined as

$$\begin{aligned}\mathbf{e}(k) &= \begin{bmatrix} d(k) \\ \lambda^{1/2} d(k-1) \\ \vdots \\ \lambda^{k/2} d(0) \end{bmatrix} - \begin{bmatrix} \mathbf{x}_P^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{x}_P^{\mathrm{T}}(k-1) \\ \vdots \\ \lambda^{k/2}\mathbf{x}_P^{\mathrm{T}}(0) \end{bmatrix} \mathbf{w}_P(k) \\ &= \mathbf{d}(k) - \mathbf{X}_P(k)\mathbf{w}_P(k), \end{aligned} \tag{6.2}$$

and

$$\mathbf{x}_P^{\mathrm{T}}(k) = \begin{bmatrix} \mathbf{x}_k^{\mathrm{T}} & \mathbf{x}_{k-1}^{\mathrm{T}} & \cdots & \mathbf{x}_{k-N+1}^{\mathrm{T}} \end{bmatrix}, \tag{6.3}$$

where $\mathbf{x}_k^{\mathrm{T}} = [x_1(k) \quad x_2(k) \quad \cdots \quad x_M(k)]$ is the input signal vector at instant $k$. As illustrated in Figure 6.1, $N$ is the number of filter coefficients per channel, $M$ is the number of input channels, and $\mathbf{w}_P(k)$ is the $P \times 1$ coefficient vector at time instant $k$, $P = MN$ being the total number of elements for the case of channels with equal orders.

The good numerical behavior of the QR-decomposition-based RLS algorithms is due to the fact that they make use of the square root $\mathbf{U}_P(k)$ of the input-data autocorrelation matrix $\mathbf{X}_P^T(k)\mathbf{X}_P(k)$. The lower triangular matrix $\mathbf{U}_P(k)$ is also known as the Cholesky factor of $\mathbf{X}_P^T(k)\mathbf{X}_P(k)$ and can be obtained by applying a set of Givens rotations $\mathbf{Q}_P(k)$ onto $\mathbf{X}_P(k)$. Hence, pre-multiplying both sides of (6.2) by unitary matrix $\mathbf{Q}_P(k)$ does not alter its norm yielding

$$\mathbf{e}_q(k) = \mathbf{Q}_P(k)\mathbf{e}(k) = \begin{bmatrix} \mathbf{e}_{q1}(k) \\ \mathbf{e}_{q2}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{d}_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \mathbf{U}_P(k) \end{bmatrix} \mathbf{w}_P(k). \tag{6.4}$$

Again, minimizing $\|\mathbf{e}_q(k)\|^2$ is equivalent to minimizing the cost function of (6.1). In other words, Equation (6.1) is minimized by choosing $\mathbf{w}_P(k)$ in (6.4) such that $\mathbf{d}_{q2}(k) - \mathbf{U}_P(k)\mathbf{w}_P(k)$ equals zero, i.e.,

$$\mathbf{w}_P(k) = \mathbf{U}_P^{-1}(k)\mathbf{d}_{q2}(k). \tag{6.5}$$

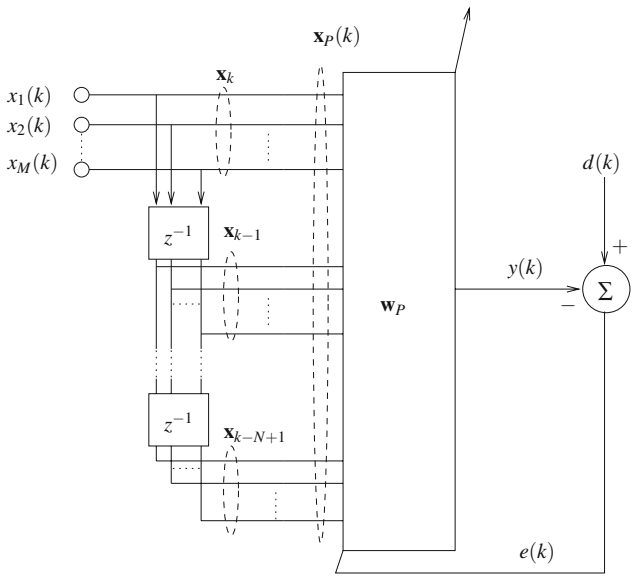


**Fig. 6.1** Multichannel adaptive filter: case of equal order.

In many applications, we may come across the need of dealing with different channel orders which is referred to as the multiple-order case. In such a scenario, the elements of the input vector are arranged in a slightly different manner than for the equal-order case treated above. Below we explain how to build the input vector such that we can cope with both equal- and multiple-order channels.

### 6.2.1 Redefining the input vector

Let us define $N_1, N_2, \ldots, N_M$ as the number of *taps* in each of the $M$ channels *tapped delay lines*. Thus, the total number of taps in the input vector is $P = \sum_{r=1}^{M} N_r$. Without loss of generality, we will assume $N_1 \geq N_2 \geq \cdots \geq N_M$.

Figure 6.2 shows an example of a multichannel scenario with $M = 3$ channels of unequal orders, where $N_1 = 4, N_2 = 3, N_3 = 2$, i.e., $P = 4+3+2 = 9$. The following approach to construct the input vector, $\mathbf{x}_P(k)$, was considered in [11] first channel are chosen to be the leading elements of $\mathbf{x}_P(k)$, followed by $N_2$–$N_3$ *pairs of samples* from the first and second channels, followed by $N_3$–$N_4$ *triples of samples* of the first three channels and so on till the $N_M$–$N_{M+1}$ *M-tuples of samples* of all channels. It is assumed that $N_{M+1} = 0$.

Alternatively, consider the $N_1 \times M$ matrix $\tilde{\mathbf{X}}(k)$ whose $i$th row contains the $N_i$ input data samples of channel $i$, i.e.,

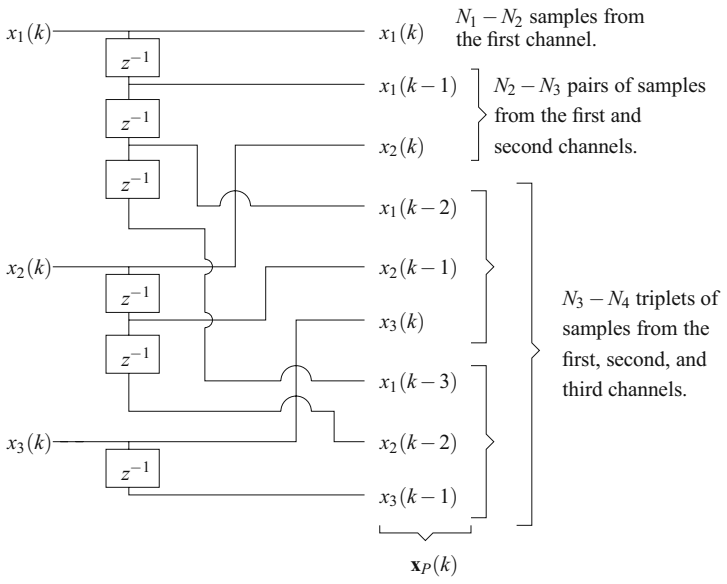


**Fig. 6.2** Example of input vector defined as in [11].

$$\tilde{\mathbf{X}}(k) = \begin{bmatrix} x_1(k) & x_1(k-1) & x_1(k-2) & \dots & x_1(k-N_1+1) \\ \mathbf{0}_{1\times(N_1-N_2)} & x_2(k) & x_2(k-1) & \dots & x_2(k-N_2+1) \\ \vdots & & \vdots & & \vdots \\ \mathbf{0}_{1\times(N_1-N_M)} & & x_M(k) & \dots & x_M(k-N_M+1) \end{bmatrix} \tag{6.6}$$

where the zero-vectors appearing to the left in each row are of proper size to maintain the dimension of $\tilde{\mathbf{X}}(k)$ (if $N_1 = N_2 = \cdots = N_M$, the zeros will disappear). The multichannel input vector $\mathbf{x}_P(k)$ is obtained by simply stacking the columns of matrix $\tilde{\mathbf{X}}(k)$ and excluding the zeros that were inserted in (6.6). We see from (6.6) that the last $M$ samples of vector $\mathbf{x}_P(k)$ are $\{x_l(k-N_i)\}_{i=1}^{M}$. As a result, updating the input vector from one time instant to the next, i.e., from $\mathbf{x}_P(k)$ to $\mathbf{x}_P(k+1)$, becomes particularly simple. This is because we know that, by construction, the last $M$ samples of $\mathbf{x}_P(k)$ are to be removed when shifting in the new input samples $\{x_i(k+1)\}_{i=1}^{M}$.

The procedure detailed above gives rise to two distinct ways of obtaining the expanded input vector, $\mathbf{x}_{P+M}(k+1)$: (1) The new samples from each channel are shifted one-by-one and processed recursively, from the first to the last channel and; (2) All new samples from the different channels are shifted in together and processed simultaneously.

The first approach leads to sequential-type multichannel algorithms and the second one results in block-type multichannel algorithms. Before presenting the different algorithms, we take a closer look at the sequential- and block-type input vectors.

### 6.2.2 Input vector for sequential-type multichannel algorithms

For the sequential-channel case, the extended input vector, $\mathbf{x}_{P+M}(k+1)$, is constructed from $\mathbf{x}_P(k)$ in $M$ successive steps as

$$\mathbf{x}_{P+1}^{\mathrm{T}}(k+1) = \begin{bmatrix} x_1(k+1) & \mathbf{x}_P^{\mathrm{T}}(k) \end{bmatrix}, \tag{6.7}$$

$$\mathbf{x}_{P+i}^{\mathrm{T}}(k+1) = \begin{bmatrix} x_i(k+1) & \mathbf{x}_{P+i-1}{}^{\mathrm{T}}(k+1) \end{bmatrix} \mathbf{P}_i, \tag{6.8}$$

where $\mathbf{P}_i$ is a permutation matrix which takes the most recent sample $x_i(k+1)$ of the $i$th channel to position $p_i$ and left shifts the first $p_i - 1$ elements of $\mathbf{x}_{P+i-1}^{\mathrm{T}}(k+1)$, where

$$p_i = \sum_{r=1}^{i-1} r(N_r - N_{r+1}) + i, \;\; i = 1, 2, \ldots, M. \tag{6.9}$$

After processing all $M$ channels, the first $P$ elements of the updated extended input vector constitute the input vector of the next iteration, i.e., $\mathbf{x}_{P+M}^{\mathrm{T}}(k+1) = [\mathbf{x}_P^{\mathrm{T}}(k+1) \quad x_1(k-N_1+1) \quad \cdots \quad x_M(k-N_M+1)]$.

### 6.2.3 Input vector for block-type multichannel algorithms

For the case of block-channel multichannel algorithms, the expanded input vector, $\mathbf{x}_{P+M}(k+1)$, is given by

$$\begin{aligned}\mathbf{x}_{P+M}^{\mathrm{T}}(k+1) &= \begin{bmatrix} x_1(k+1) & x_2(k+1) & \cdots & x_M(k+1) & \mathbf{x}_P^{\mathrm{T}}(k)\end{bmatrix}\mathbf{P} \\ &= \begin{bmatrix}\mathbf{x}_{k+1}^{\mathrm{T}} & \mathbf{x}_P^{\mathrm{T}}(k)\end{bmatrix}\mathbf{P}, \end{aligned} \tag{6.10}$$

where $\mathbf{P} = \mathbf{P}_M\mathbf{P}_{M-1}\cdots\mathbf{P}_1$ is a product of $M$ permutation matrices that moves the most recent sample of the $i$th channel (for $i = 1,2,\ldots,M$) to position $p_i$ (given by Equation (6.9)) in vector $\mathbf{x}_{P+M}(k+1)$.

After the above process is terminated, we have $\mathbf{x}_{P+M}^{\mathrm{T}}(k+1) = [\mathbf{x}_P^{\mathrm{T}}(k+1) \quad x_1(k-N_1+1)\cdots \quad x_M(k-N_M+1)]$, such that the first $P$ elements of $\mathbf{x}_{P+M}^{\mathrm{T}}(k+1)$ provide the input vector for the next iteration. In order to illustrate the role of the permutation matrix $\mathbf{P}$, let us return to the example depicted in Figure 6.2. In this example, the expanded input vector $\mathbf{x}_{P+M}(k+1)$ is obtained by inserting the new samples in positions $p_1 = 1$, $p_2 = 3$, and $p_3 = 6$, respectively, i.e.,

$$\mathbf{P}^{\mathrm{T}}\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ \underline{x_3(k+1)} \\ \mathbf{x}_P(k)\end{bmatrix} = \mathbf{P}^{\mathrm{T}}\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ \underline{x_3(k+1)} \\ x_1(k) \\ x_1(k-1) \\ x_2(k) \\ x_1(k-2) \\ x_2(k-1) \\ x_3(k) \\ x_1(k-3) \\ x_2(k-2) \\ x_3(k-1)\end{bmatrix} = \begin{bmatrix} x_1(k+1) \\ x_1(k) \\ x_2(k+1) \\ x_1(k-1) \\ x_2(k) \\ x_3(k+1) \\ x_1(k-2) \\ x_2(k-1) \\ \underline{x_3(k)} \\ x_1(k-3) \\ x_2(k-2) \\ x_3(k-1)\end{bmatrix} = \begin{bmatrix}\mathbf{x}_P(k+1) \\ \underline{x_1(k-3)} \\ x_2(k-2) \\ x_3(k-1)\end{bmatrix}. \tag{6.11}$$

## 6.3 Sequential-Type MC-FQRD-RLS Algorithms

In this section, we consider algorithms that process the channels sequentially. In the following, we shall derive the *a priori* [11] and the *a posteriori* [3] versions of sequential MC-FQRD-RLS algorithms based on updating backward error vectors.

Due to close similarities with the single-channel case, basic concepts on how to solve the *backward* and *forward* prediction problems are omitted. Indeed, the sequential processing of multichannel signals corresponds to solving the single-channel algorithm $M$ times, with $M$ being the number of channels. Moreover, from the forward prediction problem, the extended input data matrices used in sequential-channel algorithms are defined as

$$\mathbf{X}_{P+i}(k) = \begin{bmatrix} \mathbf{x}_{P+i}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{x}_{P+i}^{\mathrm{T}}(k-1) \\ \vdots \\ \lambda^{k/2}\mathbf{x}_{P+i}^{\mathrm{T}}(0) \end{bmatrix}, \quad i = 1, 2, \ldots, M, \tag{6.12}$$

where vector $\mathbf{x}_{P+i}(k)$ is the extended input vector defined in Equation (6.8).

### 6.3.1 Triangularization of the information matrix

Equation (6.12) suggests that the updating of the information matrix is performed in $M$ forward steps for each iteration.

**First step** ($i = 1$)

$\mathbf{X}_{P+1}(k)$ can be defined as

$$\mathbf{X}_{P+1}(k) = \begin{bmatrix} x_1(k) & \\ \lambda^{1/2}x_1(k-1) & \mathbf{X}_P(k-1) \\ \vdots & \\ \lambda^{k/2}x_1(0) & \mathbf{0}^{\mathrm{T}} \end{bmatrix} = \left[ \mathbf{d}_f^{(1)}(k) \middle| \begin{matrix} \mathbf{X}_P(k-1) \\ \mathbf{0}^{\mathrm{T}} \end{matrix} \right], \tag{6.13}$$

where $\mathbf{d}_{f1}^{(1)}(k) = [x_1(k) \quad \lambda^{1/2}x_1(k-1) \quad \cdots \quad \lambda^{k/2}x_1(0)]$.

Let $\mathbf{Q}_P^{(1)}(k)$ be the orthogonal matrix associated with the Cholesky factor $\mathbf{U}_P(k-1)$ of matrix $\mathbf{X}_P^{\mathrm{T}}(k-1)\mathbf{X}_P(k-1)$. Then, from (6.13), we can write

$$\begin{bmatrix} \mathbf{Q}_P^{(1)}(k) & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{1\times 1} \end{bmatrix} \left[ \mathbf{d}_f^{(1)}(k) \middle| \begin{matrix} \mathbf{X}_P(k-1) \\ \mathbf{0}^{\mathrm{T}} \end{matrix} \right] = \begin{bmatrix} \mathbf{e}_{fq1}^{(1)}(k) & \mathbf{0} \\ \mathbf{d}_{fq2}^{(1)}(k) & \mathbf{U}_P(k-1) \\ \lambda^{k/2}x_1(0) & \mathbf{0}^{\mathrm{T}} \end{bmatrix}. \tag{6.14}$$

To complete the triangularization process of $\mathbf{X}_{P+1}(k)$ leading to $\mathbf{U}_{P+1}(k)$, we pre-multiply (6.14) by two other Givens rotation matrices as follows:

$$\begin{aligned} \begin{bmatrix} \mathbf{0} \\ \mathbf{U}_{P+1}(k) \end{bmatrix} &= \mathbf{Q}_f'^{(1)}(k)\mathbf{Q}_f{}^{(1)}(k) \begin{bmatrix} \mathbf{e}_{fq1}^{(1)}(k) & \mathbf{0} \\ \mathbf{d}_{fq2}^{(1)}(k) & \mathbf{U}_P(k-1) \\ \lambda^{k/2}x_1(0) & \mathbf{0}^{\mathrm{T}} \end{bmatrix} \\ &= \mathbf{Q}_f'^{(1)}(k) \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{d}_{fq2}^{(1)}(k) & \mathbf{U}_P(k-1) \\ e_{fP}^{(1)}(k) & \mathbf{0}^{\mathrm{T}} \end{bmatrix}. \end{aligned} \tag{6.15}$$

In the previous equation, $\mathbf{Q}_f{}^{(1)}(k)$ is the orthogonal matrix zeroing $\mathbf{e}_{fq1}^{(1)}(k)$ generating $e_{fP}^{(1)}(k)$. Matrix $\mathbf{Q}'_f{}^{(1)}(k)$ completes the triangularization process by zeroing $\mathbf{d}_{fq2}^{(1)}(k)$ from (6.15) in a top down procedure against $e_{fP}^{(1)}(k)$. Removing the resulting null section in the upper part of (6.15) gives

$$\mathbf{U}_{P+1}(k) = \mathbf{Q}'_{\theta f}{}^{(1)}(k) \begin{bmatrix} \mathbf{d}_{fq2}^{(1)}(k) & \mathbf{U}_P(k-1) \\ e_{fP}^{(1)}(k) & \mathbf{0}^{\mathrm{T}} \end{bmatrix}. \tag{6.16}$$

From (6.16), we get the following relation that is useful for the updating of $\mathbf{a}_P(k)$ and $\mathbf{f}_P(k)$, the *a priori* and the *a posteriori* error vectors, respectively.

$$[\mathbf{U}_{P+1}(k+1)]^{-1} = \begin{bmatrix} \mathbf{0}^{\mathrm{T}} & \frac{1}{e_{fP}^{(1)}(k+1)} \\ \mathbf{U}_P^{-1}(k) & -\frac{1}{e_{fP}^{(1)}(k+1)}\mathbf{U}_P^{-1}(k)\mathbf{d}_{fq2}^{(1)}(k+1) \end{bmatrix} \left[\mathbf{Q}'_{\theta f}{}^{(1)}(k+1)\right]^{\mathrm{T}} \tag{6.17}$$

Also from (6.16), we see that $\mathbf{Q}'_{\theta f}{}^{(1)}(k)$ is the Givens rotation matrix responsible for zeroing $\mathbf{d}_{fq2}^{(1)}(k)$ against $e_{fP}^{(1)}(k)$, i.e.,

$$\begin{bmatrix} \mathbf{0} \\ e_{f0}^{(1)}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}{}^{(1)}(k+1) \begin{bmatrix} \mathbf{d}_{fq2}^{(1)}(k+1) \\ e_{fP}^{(1)}(k+1) \end{bmatrix}. \tag{6.18}$$

The updating of $\mathbf{d}_{fq2}^{(1)}(k)$ is performed according to

$$\begin{bmatrix} \tilde{e}_{fq1}^{(1)}(k+1) \\ \mathbf{d}_{fq2}^{(1)}(k+1) \end{bmatrix} = \mathbf{Q}_{\theta_P}^{(0)}(k) \begin{bmatrix} x_1(k+1) \\ \lambda^{1/2}\mathbf{d}_{fq2}^{(1)}(k) \end{bmatrix}, \tag{6.19}$$

where $\mathbf{Q}_{\theta_P}^{(0)}(k) = \mathbf{Q}_{\theta_P}^{(M)}(k-1)$, i.e., the values obtained after processing the $M$th channel on last iteration. For $1 < i \leq M$, $\mathbf{d}_{fq2}^{(i)}(k)$ is updated according to

$$\begin{bmatrix} \tilde{e}_{fq1}^{(i)}(k+1) \\ \mathbf{d}_{fq2}^{(i)}(k+1) \end{bmatrix} = \mathbf{Q}_{\theta_{P+i-1}}^{(i-1)}(k+1) \begin{bmatrix} x_i(k+1) \\ \lambda^{1/2}\mathbf{d}_{fq2}^{(i)}(k) \end{bmatrix}. \tag{6.20}$$

**Following steps ($i > 1$)**

The input information matrix $\mathbf{X}_{P+i}(k)$ is related to $\mathbf{X}_{P+i-1}(k)$ according to

$$\mathbf{X}_{P+i}(k) = \begin{bmatrix} x_i(k) \\ \lambda^{1/2} x_i(k-1) \\ \vdots \\ \lambda^{k/2} x_i(0) \end{bmatrix} \quad \mathbf{X}_{P+i-1}(k) \Bigg] \mathbf{P}_i. \tag{6.21}$$

As in the first step, matrix $\mathbf{X}_{P+i}(k)$ must be triangularized to obtain $\mathbf{U}_{P+i}(k)$ (Cholesky factor of $\mathbf{X}_{P+i}^{\mathrm{T}}(k)\mathbf{X}_{P+i}(k)$). This process is detailed in the following. Let $\mathbf{Q}_{\theta_{P+i-1}}(k)$ denotes the orthogonal matrix associated with the QR decomposition of $\mathbf{X}_{P+i-1}(k)$. From (6.21), we can write

$$\begin{aligned} &\mathbf{Q}_f^{(i)}(k) \begin{bmatrix} \mathbf{Q}_{P+i-1}(k) & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X}_{P+i}(k) \\ \mathbf{0}^{\mathrm{T}} \end{bmatrix} = \\ &\mathbf{Q}_f^{(i)}(k) \begin{bmatrix} \mathbf{e}^{(i)}_{fq1_{P+i-1}}(k) & \mathbf{0} \\ \mathbf{d}^{(i)}_{fq2}(k) & \mathbf{U}_{P+i-1}(k) \\ \lambda^{k/2}\mathbf{x}_i(0) & \mathbf{0}^{\mathrm{T}} \end{bmatrix} \mathbf{P}_i = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{d}^{(i)}_{fq2}(k) & \mathbf{U}_{P+i-1}(k) \\ e^{(i)}_{f_{P+i-1}}(k) & \mathbf{0}^{\mathrm{T}} \end{bmatrix} \mathbf{P}_i. \end{aligned} \tag{6.22}$$

Equation (6.22) is obtained by annihilating $\mathbf{e}^{(i)}_{fq1_{P+i-1}}(k)$ into the first element of the last row of the matrix using an appropriate orthogonal matrix, $\mathbf{Q}_f^{(i)}(k)$, and thereafter removing the resulting null section.

The existence of the permutation matrix $\mathbf{P}_i$ in (6.22) prevents us from directly annihilating $\mathbf{d}^{(i)}_{fq2}(k)$ into $e^{(i)}_{f_{P+i-1}}(k)$ to complete the triangularization of matrix $\mathbf{X}_{P+i}(k)$ (i.e., generating $\mathbf{U}_{P+i}(k)$). Figure 6.3 illustrates the application of Givens rotations under these circumstances. The process is summarized as follows. The permutation factor, $\mathbf{P}_i$, right shifts $\mathbf{d}^{(i)}_{fq2}(k)$ to the $i$th position as shown in the first part of the figure. Then, the set of $P+i-p_i$ Givens rotation matrices, $\mathbf{Q}'^{(i)}_{\theta f}$, nullifies the first $P+i-p_i$ elements of $\mathbf{d}^{(i)}_{fq2}(k)$ by rotating them against $e^{(i)}_{f_{P+i-1}}(k)$ in a top down procedure. The desired triangular structure is finally reached using another permu-

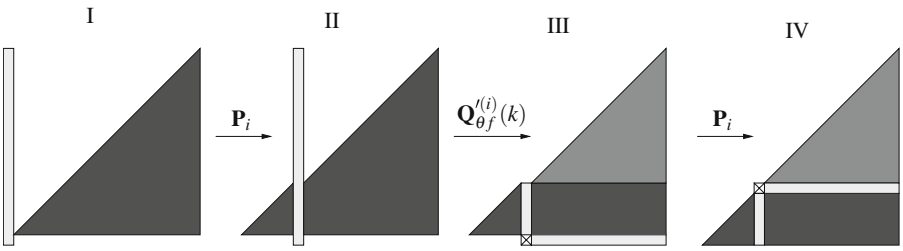


**Fig. 6.3** Obtaining the lower triangular factor $\mathbf{U}_{P+i}(k)$. The lighter color tone on top of parts III and IV represents the matrix elements that have been rotated against the bottom line by the set of Givens rotations in $\mathbf{Q}'^{(i)}_{\theta f}(k)$.

tation factor that moves the last row of the matrix to the $P-p_i+1$ position, after downshifting the previous $P-p_i$ rows. This permutation factor coincides with $\mathbf{P}_i$.

*Remark 1.* The lower triangular matrix $\mathbf{U}_{P+i}(k)$, obtained as described above must be positive definite. That is guaranteed if its diagonal elements and $e^{(i)}_{f_{P+i-1}}(k)$ are positive. Recalling that $e^{(i)}_{f_{P+i-1}}(k)$ is the absolute value of the forward error, $\mathbf{U}_{P+i}(k)$ will be positive definite if it is initialized properly.

The procedure above can be written in a more compact form as

$$\mathbf{U}_{P+i}(k) = \mathbf{P}_i \mathbf{Q}'_{\theta f}{}^{(i)}(k) \begin{bmatrix} \mathbf{d}^{(i)}_{fq2}(k) & \mathbf{U}_{P+i-1}(k) \\ e^{(i)}_{f_{P+i-1}}(k) & \mathbf{0}^{\mathrm{T}} \end{bmatrix} \mathbf{P}_i. \tag{6.23}$$

From (6.23), the following relation can be derived

$$\begin{aligned} &[\mathbf{U}_{P+i}(k+1)]^{-1} = \mathbf{P}_i^{\mathrm{T}} \\ &\times \begin{bmatrix} \mathbf{0}^{\mathrm{T}} & \dfrac{1}{e^{(i)}_{f_{P+i-1}}(k+1)} \\ \mathbf{U}^{-1}_{P+i-1}(k+1) & -\dfrac{\mathbf{U}^{-1}_{P+i-1}(k+1)\mathbf{d}^{(i)}_{fq2}(k+1)}{e^{(i)}_{f_{P+i-1}}(k+1)} \end{bmatrix} \times \mathbf{Q}'^{\mathrm{T}}_{\theta f}{}^{(i)}(k+1)\mathbf{P}_i^{\mathrm{T}}. \end{aligned} \tag{6.24}$$

From (6.23), we realize that $\mathbf{Q}'_{\theta f}{}^{(i)}(k)$ is the Givens rotation matrix responsible for zeroing $\mathbf{d}^{(i)}_{fq2}(k)$ against $e^{(i)}_{fP}(k)$, which yields

$$\begin{bmatrix} \mathbf{0} \\ e^{(i)}_{f0}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}{}^{(i)}(k+1) \begin{bmatrix} \mathbf{d}^{(i)}_{fq2}(k+1) \\ e^{(i)}_{fP}(k+1) \end{bmatrix}. \tag{6.25}$$

### 6.3.2 A priori and A posteriori versions

The *a priori* and the *a posteriori* versions of the sequential-channel algorithm are based on updating expanded vectors $\mathbf{a}_{P+i}(k+1)$ or $\mathbf{f}_{P+i}(k+1)$ given by

$$\mathbf{a}_{P+i}(k+1) = \lambda^{-1/2}\mathbf{U}^{-T}_{P+i}(k+1)\mathbf{x}_{P+i}(k+1) = \begin{bmatrix} \mathbf{a}^{(i)}(k+1) \\ \mathbf{a}_P(k+1) \end{bmatrix}, \quad \text{and} \tag{6.26}$$

$$\mathbf{f}_{P+i}(k+1) = \mathbf{U}^{-T}_{P+i}(k+1)\mathbf{x}_{P+i}(k+1) = \begin{bmatrix} \mathbf{f}^{(i)}(k+1) \\ \mathbf{f}_P(k+1) \end{bmatrix}, \tag{6.27}$$

for $i = 1, 2, \ldots, M$.

In (6.26) and (6.27), $\mathbf{a}^{(i)}(k+1)$ and $\mathbf{f}^{(i)}(k+1)$ are vectors containing the first $i$ elements of $\mathbf{a}_{P+i}(k+1)$ $\mathbf{f}_{P+i}(k+1)$, respectively. Recall that for $i=1$, $\mathbf{U}_{P+i}^{-1}(k+1)$ in (6.26) and (6.27) equals $\mathbf{U}_{P+1}^{-1}(k)$ as defined in (6.17).

The updating of $\mathbf{a}_{P+i}(k+1)$ and $\mathbf{f}_{P+i}(k+1)$ is accomplished in $M$ forward steps at each instant $k$:

$$\begin{aligned}\mathbf{a}_P(k) &\rightarrow \mathbf{a}_{P+1}(k+1) \rightarrow \ \cdots \ \rightarrow \mathbf{a}_{P+M}(k+1),\\ \mathbf{f}_P(k) &\rightarrow \mathbf{f}_{P+1}(k+1) \rightarrow \ \cdots \ \rightarrow \mathbf{f}_{P+M}(k+1).\end{aligned}$$

From (6.24), (6.8), and (6.27), we get the following recursive expression for $\mathbf{f}_{P+i}(k+1)$:

$$\mathbf{f}_{P+i}(k+1) = \mathbf{P}_i \mathbf{Q}'_{\theta f}{}^{(i)}(k+1) \begin{bmatrix} \mathbf{f}_{P+i-1}(k+1) \\ p_{P+i-1}^{(i)}(k+1) \end{bmatrix}, \tag{6.28}$$

where

$$p_{P+i-1}^{(i)}(k+1) = \frac{e_{P+i-1}^{(i)}(k+1)}{|e_{f_{P+i-1}}^{(i)}(k+1)|}, \quad \text{for } i=1,2,\ldots,M. \tag{6.29}$$

The scalar quantity $e_{P+i-1}^{(i)}(k+1)$ is the *a posteriori* forward prediction error for the $i$th channel, and $|e_{f_{P+i-1}}^{(i)}(k+1)|$ is given by

$$|e_{f_{P+i-1}}^{(i)}(k+1)| = \sqrt{\left(\lambda^{1/2}|e_{f_{P+i-1}}^{(i)}(k)|\right)^2 + |e_{fq1_{P+i-1}}^{(i)}(k+1)|^2}. \tag{6.30}$$

For $i=1$, rather than (6.8), we would use (6.7) in obtaining (6.28), which simply means that $\mathbf{P}_1 = \mathbf{I}$, i.e,

$$\mathbf{f}_{P+1}(k+1) = \mathbf{Q}'_{\theta f}{}^{(1)}(k+1) \begin{bmatrix} \mathbf{f}_P(k) \\ p^{(1)}(k+1) \end{bmatrix}, \tag{6.31}$$

with $e_P{}^{(1)}(k+1)$ denoting the *a posteriori* forward prediction error of the first channel, and $|e_{f_P}^{(1)}(k+1)|$ is given by

$$|e_{f_P}^{(1)}(k+1)| = \sqrt{\left(\lambda^{1/2}|e_{f_P}^{(1)}(k)|\right)^2 + |(e_{fq1_P}^{(i)}(k+1)|^2}. \tag{6.32}$$

Similarly, using (6.24), (6.8), and (6.26), we get the following recursive expression for $\mathbf{a}_{P+i}(k+1)$:

$$\mathbf{a}_{P+i}(k+1) = \lambda^{-1/2}\mathbf{P}_i \mathbf{Q}'_{\theta f}{}^{(i)}(k+1) \begin{bmatrix} \mathbf{a}_{P+i-1}(k+1) \\ r_{P+i-1}^{(i)}(k+1) \end{bmatrix}, \tag{6.33}$$

where

$$r_{P+i-1}^{(i)}(k+1) = \frac{e_{P+i-1}^{\prime(i)}(k+1)}{\gamma^{(i-1)}\sqrt{\lambda}|e_{f_{P+i-1}}^{(i)}(k)|}, \quad \text{for } i = 1,2,\ldots,M, \tag{6.34}$$

and scalar quantity $e_{P+i-1}^{(i)}(k)$ is the *a priori* forward prediction error for the $i$th channel. Again, for $i = 1$, we use (6.7) in obtaining (6.33) instead of (6.8), yielding

$$\mathbf{a}_{P+1}(k+1) = \lambda^{-1/2}\mathbf{Q}^{\prime}{}_{\theta f}^{(1)}(k+1)\begin{bmatrix}\mathbf{a}_P(k)\\ r^{(1)}(k+1)\end{bmatrix}. \tag{6.35}$$

*Remark 2.* Examining (6.28) and recalling the definitions of $\mathbf{Q}^{\prime}{}_{\theta f}^{(i)}(k+1)$ and $\mathbf{P}_i$, we realize that the last $p_i - 1$ elements of $\mathbf{f}_{P+i}(k+1)$ and $\mathbf{f}_{P+i-1}(k+1)$ are identical. Indeed, Givens rotations in $\mathbf{Q}^{\prime}{}_{\theta f}^{(i)}(k+1)$ are such that they act on a smaller portion of $\mathbf{f}_{P+i}(k+1)$ at each $i$; then $\mathbf{P}_i$ shifts down the unchanged elements which remain unchanged for next $i$. This fact reduces the computational cost. The same observation holds for vectors $\mathbf{a}_{P+i}(k+1)$ and $\mathbf{a}_{P+i-1}(k+1)$.

For the algorithm based on updating the *a posteriori* backward errors, the Givens rotations matrices $\mathbf{Q}_{\theta_{P+i}}(k+1)$ needed in the next forward steps are obtained from

$$\mathbf{Q}_{\theta_{P+i}}^{(i)}(k+1)\begin{bmatrix}1\\ \mathbf{0}\end{bmatrix} = \begin{bmatrix}\gamma_{P+i}^{(i)}(k+1)\\ \mathbf{f}_{P+i}(k+1)\end{bmatrix}, \quad \text{for } i \geq 1. \tag{6.36}$$

For the *a priori* case, the equivalent relations are

$$\begin{bmatrix}1/\gamma_{P+i}^{(i)}(k+1)\\ \mathbf{0}\end{bmatrix} = \mathbf{Q}_{\theta_{P+i}}^{(i)}(k+1)\begin{bmatrix}1\\ \mathbf{a}_{P+i}(k+1)\end{bmatrix}, \quad \text{for } i \geq 1. \tag{6.37}$$

After the $M$th channel is processed, the joint process estimation is performed according to

$$\begin{bmatrix}e_{q1}(k+1)\\ \mathbf{d}_{q2}(k+1)\end{bmatrix} = \mathbf{Q}_{\theta}^{(0)}(k+1)\begin{bmatrix}e_{q1}(k)\\ \mathbf{d}_{q2}(k)\end{bmatrix}. \tag{6.38}$$

The *a posteriori* and the *a priori* multiple order sequential algorithms are summarized in Tables 6.2 and 6.3, respectively.

### *6.3.3 Alternative implementations*

As observed before, when all channels are of equal order, matrix $\mathbf{P}_i$ in (6.23) degenerates to identity and, after carrying out the multiplication by $\mathbf{Q}^{\prime}{}_{\theta f}^{(i)}(k)$ on the right side of that equation, $\mathbf{U}_{P+i}(k)$ can be partitioned as follows.

**Table 6.2** Algorithm number **8** of Table 6.1 [10].

The multiple order sequential-type MC-FQRD_POS_B

Initializations:

$\mathbf{d}_{fq2}^{(i)} = zeros(P,1)$; $\mathbf{f}_j^{(M)}(0) = \mathbf{0}$; $\mathbf{d}_{q2} = \mathbf{0}$; $\gamma_P^{(0)}(0) = 1$;

$e_{fP}^{(i)}(0) = \mu$; $i = 1,2,\cdots,M$, all cosines = 1, and all sines = 0.

for $k = 1,2,\cdots$

{ $\gamma_0^{(1)} = 1$; $e_{q1}^{(0)}(k+1) = d(k+1)$;

for $i = 1 : M$,

{ $e_{fq1_0}^{(i)}(k+1) = x_i(k+1)$;

for $j = 1 : P$, % Obtaining $e_{fq1}^{(i)}(k+1)$ and $\mathbf{d}_{fq2}^{(i)}(k+1)$:

$\{e_{fq1_j}^{(i)}(k+1) = \cos\left[\theta_j^{(i-1)}(k)\right] e_{fq1_{j-1}}^{(i)}(k+1) + \lambda^{1/2}\sin\left[\theta_j^{(i-1)}(k)\right]\mathbf{d}_{fq2_{P-j+1}}^{(i)}(k)$;

$\mathbf{d}_{fq2_{P-j+1}}^{(i)}(k+1) = \lambda^{1/2}\cos\left[\theta_j^{(i-1)}(k)\right]\mathbf{d}_{fq2_{P-j+1}}^{(i)}(k) - \sin^*\left[\theta_j^{(i-1)}(k)\right] e_{fq1_{j-1}}^{(i)}(k+1)$;

}

$\|e_{f_P}^{(i)}(k+1)\| = \sqrt{\left(\lambda^{1/2}\|e_{f_P}^{(i)}(k)\|\right)^2 + \|e_{fq1_P}^{(i)}(k+1)\|^2}$;

for $j = P : -1 : p_i$, % Obtaining $\mathbf{Q}'^{(i)}_{\theta f}(k+1)$:

$\{\|e_{f_{j-1}}^{(i)}(k+1)\| = \sqrt{\|e_{f_j}^{(i)}(k+1)\|^2 + \|\mathbf{d}_{fq2_{P-j+1}}^{(i)}(k+1)\|^2}$;

$\cos\theta'^{(i)}_{f_j}(k+1) = \|e_{f_j}^{(i)}(k+1)\| / \|e_{f_{j-1}}^{(i)}(k+1)\|$;

$\sin\theta'^{(i)}_{f_j}(k+1) = \left[\cos\theta'^{(i)}_{f_j}(k+1)\ \mathbf{d}_{fq2_{P-j+1}}^{(i)}(k+1)/e_{f_j}^{(i)}(k+1)\right]^*$;

}

$p_P^{(i)}(k+1) = \gamma_P^{(i-1)}(k)\left[e_{fq1_P}^{(i)}(k+1)\right]^* / \|e_{f_P}^{(i)}(k+1)\|$;

for $j = P : -1 : p_i$, % Obtaining $\mathbf{f}^{(i)}(k+1)$:

$\{\mathbf{f}_{P-j+1}^{(i)}(k+1) = \cos\theta'^{(i)}_{f_j}(k+1)\mathbf{f}_{P-j+2}^{(i-1)}(k+1) - \left[\sin\theta'^{(i)}_{f_j}(k+1)\right]^* p_j^{(i)}(k+1)$;

$p_{j-1}^{(i)}(k+1) = \sin\theta'^{(i)}_{f_j}(k+1)\mathbf{f}_{P-j+2}^{(i-1)}(k+1) + \cos\theta'^{(i)}_{f_j}(k+1)p_j^{(i)}(k+1)$;

}

$\mathbf{f}_{P+1-p_i+1}^{(i)}(k+1) = p_{p_i-1}^{(i)}(k+1)$;

for $j = p_i : P$, % Obtaining $\mathbf{Q}_\theta^{(i)}(k)$:

$\{\sin\theta_j^{(i)}(k) = -\left[\mathbf{f}_{P-j+2}^{(i)}(k+1)\right]^* / \gamma_{j-1}^{(i)}(k)$;

$\cos\theta_j^{(i)}(k) = \sqrt{1 - \|\sin\theta_j^{(i)}(k)\|^2}$;

$\gamma_j^{(i)}(k) = \cos\theta_j^{(i)}(k)\gamma_{j-1}^{(i)}(k)$;

}

} for $i$

for $j = 1 : P$, % Joint process estimation:

$\{e_{q1}^{(j)}(k+1) = \cos\theta_j^{(0)}(k+1)e_{q1}^{(j-1)}(k+1) + \lambda^{1/2}\sin\theta_j^{(0)}(k+1)\mathbf{d}_{q2}^{(P-j+1)}(k)$;

$\mathbf{d}_{q2}^{(P-j+1)}(k+1) = \lambda^{1/2}\cos\theta_j^{(0)}(k+1)\mathbf{d}_{q2}^{(P-j+1)}(k) - \left[\sin\theta_j^{(0)}(k+1)\right]^* e_{q1}^{(j-1)}(k+1)$;

}

$\mathbf{e}(k+1) = \left[e_{q1}^{(P)}(k+1)\right]^* / \gamma_P^{(0)}(k)$;

} for $k$

Obs.: $\theta_j^{(M)}(k) = \theta_j^{(0)}(k+1)$ and $\mathbf{f}_{P-j+2}^{(M)}(k) = \mathbf{f}_{P-j+2}^{(0)}(k+1)$. The *asterisk* $(*)$ denotes complex conjugation.

$$\mathbf{U}_{P+i}(k+1) = \begin{bmatrix} \mathbf{0} & \mathbf{B} \\ e_{f0}^{(i)}(k+1) & \mathbf{C} \end{bmatrix}, \quad \text{for } i = 1,2,\ldots,M. \tag{6.39}$$

Taking the inverse of (6.39) yields

$$[\mathbf{U}_{P+i}(k+1)]^{-1} = \begin{bmatrix} -\left[e_{f0}^{(i)}(k+1)\right]^{-1}\mathbf{C}\mathbf{B}^{-1} & \left[e_{f0}^{(i)}(k+1)\right]^{-1} \\ \mathbf{B}^{-1} & \mathbf{0} \end{bmatrix}. \tag{6.40}$$

**Table 6.3** Algorithm number **16** of Table 6.1 [11].

The multiple order sequential-type MC-FQRD_PRI_B

Initializations:

$\mathbf{d}_{fq2}^{(i)} = zeros(P,1)$; $\mathbf{a}_j^{(M)}(0) = \mathbf{0}$; $\mathbf{d}_{q2} = \mathbf{0}$; $\gamma_P^{(0)}(0) = 1$;

$e_{fP}^{(i)}(0) = \mu$; $i = 1,2,\cdots,M$, all cosines = 1, and all sines = 0.

for $k = 1,2,\cdots$

{ $\gamma_0^{(1)} = 1$; $e_{q1}^{(0)}(k+1) = d(k+1)$;

for $i = 1 : M$,

{ $e_{fq1_0}^{(i)}(k+1) = x_i(k+1)$;

for $j = 1 : P$, % Obtaining $e_{fq1}^{(i)}(k+1)$ and $\mathbf{d}_{fq2}^{(i)}(k+1)$:

$\{e_{fq1_j}^{(i)}(k+1) = \cos\left[\theta_j^{(i-1)}(k)\right] e_{fq1_{j-1}}^{(i)}(k+1) + \lambda^{1/2}\sin\left[\theta_j^{(i-1)}(k)\right]\mathbf{d}_{fq2_{P-j+1}}^{(i)}(k)$;

$\mathbf{d}_{fq2_{P-j+1}}^{(i)}(k+1) = \lambda^{1/2}\cos\left[\theta_j^{(i-1)}(k)\right]\mathbf{d}_{fq2_{P-j+1}}^{(i)}(k) - \sin^*\left[\theta_j^{(i-1)}(k)\right] e_{fq1_{j-1}}^{(i)}(k+1)$;

}

$r_P^{(i)}(k+1) = \lambda^{-1/2}[e_{fq1_j}^{(i)}(k+1)]^* / \left[\gamma_P^{(i-1)}(k)\|e_{fP}^{(i)}(k)\|\right]$;

for $j = P : -1 : p_i$, % Obtaining $\mathbf{a}^{(i)}(k+1)$:

$\{\mathbf{a}_{P-j+1}^{(i)}(k+1) = \cos\theta'^{(i)}_{f_j}(k)\mathbf{a}_{P-j+2}^{(i-1)}(k+1) - \left[\sin\theta'^{(i)}_{f_j}(k)\right]^* r_j^{(i)}(k+1)$;

$r_{j-1}^{(i)}(k+1) = \sin\theta'^{(i)}_{f_j}(k)\mathbf{a}_{P-j+2}^{(i-1)}(k+1) + \cos\theta'^{(i)}_{f_j}(k) r_j^{(i)}(k+1)$;

}

$\mathbf{a}_{P+1-p_i+1}^{(i)}(k+1) = r_{p_i-1}^{(i)}(k+1)$;

$\|e_{fP}^{(i)}(k+1)\| = \sqrt{\left(\lambda^{1/2}\|e_{fP}^{(i)}(k)\|\right)^2 + \|e_{fq1P}^{(i)}(k+1)\|^2}$;

for $j = P : -1 : p_i$, % Obtaining $\mathbf{Q}'^{(i)}_{\theta f}(k+1)$:

$\{\|e_{f_{j-1}}^{(i)}(k+1)\| = \sqrt{\|e_{f_j}^{(i)}(k+1)\|^2 + \|\mathbf{d}_{fq2_{P-j+1}}^{(i)}(k+1)\|^2}$;

$\cos\theta'^{(i)}_{f_j}(k+1) = \|e_{f_j}^{(i)}(k+1)\| / \|e_{f_{j-1}}^{(i)}(k+1)\|$;

$\sin\theta'^{(i)}_{f_j}(k+1) = \left[\cos\theta'^{(i)}_{f_j}(k+1)\ \ \mathbf{d}_{fq2_{P-j+1}}^{(i)}(k+1)/\|e_{f_j}^{(i)}(k+1)\|\right]^*$;

}

for $j = p_i : P$, % Obtaining $\mathbf{Q}_\theta^{(i)}(k)$:

$\{\gamma_j^{(i)}(k) = 1/\sqrt{(1/\gamma_{j-1}^{(i)}(k))^2 + (\mathbf{a}_{P-j+2}^{(i)}(k+1))^2}$

$\cos\theta_j^{(i)}(k) = \gamma_j^{(i)}(k)/\gamma_{j-1}^{(i)}(k)$;

$\sin\theta_j^{(i)}(k) = \left[\mathbf{a}_{P-j+2}^{(i)}(k+1)\cos\theta_j^{(i)}(k)/\gamma_{j-1}^{(i)}(k)\right]^*$;

}

} for $i$

for $j = 1 : P$, % Joint process estimation:

$\{e_{q1}^{(j)}(k+1) = \cos\theta_j^{(0)}(k+1) e_{q1}^{(j-1)}(k+1) + \lambda^{1/2}\sin\theta_j^{(0)}(k+1)\mathbf{d}_{q2}^{(P-j+1)}(k)$;

$\mathbf{d}_{q2}^{(P-j+1)}(k+1) = \lambda^{1/2}\cos\theta_j^{(0)}(k+1)\mathbf{d}_{q2}^{(P-j+1)}(k) - \left[\sin\theta_j^{(0)}(k+1)\right]^* e_{q1}^{(j-1)}(k+1)$;

}

$\mathbf{e}(k+1) = \left[e_{q1}^{(P)}(k+1)\right]^* / \gamma_P^{(0)}(k+1)$;

} for $k$

Obs.: $\theta_j^{(M)}(k) = \theta_j^{(0)}(k+1)$ and $\mathbf{a}_{P-j+2}^{(M)}(k) = \mathbf{a}_{P-j+2}^{(0)}(k+1)$. The *asterisk* $(*)$ denotes complex conjugation.

Using (6.40), and recalling (6.26), (6.27), and the definition of the input vector given in (6.8), we can now write vectors $\mathbf{a}_{P+i}(k+1)$ and $\mathbf{f}_{P+i}(k+1)$, respectively, as

$$\mathbf{a}_{P+i}(k+1) = \begin{bmatrix} \\ x_i(k+1)/\left[\sqrt{\lambda}e_{f0}^{(i)}(k)\right]^* \end{bmatrix}, \text{ and} \tag{6.41}$$

$$\mathbf{f}_{P+i}(k+1) = \begin{bmatrix} \\ x_i(k+1)/e_{f0}^{(i)}(k). \end{bmatrix}^* \tag{6.42}$$

As we can see from (6.41) and (6.42), the last element of $\mathbf{a}_{P+i}(k+1)$ and $\mathbf{f}_{P+i}(k+1)$ are known at each iteration $i$ (for $i = 1,2,\ldots,M$) prior to the updating process. That observation is the key step leading to two alternative implementations of these algorithms for the special case when the channels are of the same order. Thus, the recursive updating of vectors $\mathbf{a}_{P+i}(k+1)$ and $\mathbf{f}_{P+i}(k+1)$ are performed now based on this assumption.

## 6.4 Block-Type MC-FQRD-RLS Algorithms

This section discusses a general framework for block-type multichannel algorithms using the extended input signal vector $\mathbf{x}_{P+M}(k+1)$ defined in Section 6.2.3. These algorithms, despite exhibiting a higher computational burden as compared to the sequential-type ones, have some attractive features, e.g., suitability for parallel implementation.

To begin with, in Section 6.4.1 we shall revisit the backward and forward prediction problems applied to a block-multichannel scenario from where some fundamental equations are derived. The *a priori* and the *a posteriori* versions are discussed in Section 6.4.2 followed by a brief overview of some alternative implementations in Section 6.4.3.

### *6.4.1 The backward and forward prediction problems*

Using the definition of the input vector for the multiple order channels case as given by (6.10), define the input data matrix $\mathbf{X}_{P+M}(k+1)$ as follows.

$$\mathbf{X}_{P+M}(k+1) = \begin{bmatrix} \mathbf{x}_{P+M}^{\mathrm{T}}(k+1) \\ \lambda^{1/2}\mathbf{x}_{P+M}^{\mathrm{T}}(k) \\ \vdots \\ \lambda^{(k+1)/2}\mathbf{x}_{P+M}^{\mathrm{T}}(0) \end{bmatrix} \tag{6.43}$$

The above matrix can be partitioned into two distinct ways, depending onto the prediction problem, **backward** or **forward**, to be solved.

Define the **backward prediction** error matrix for block processing of equal order channels as

$$\begin{aligned} \mathbf{E}^{b}(k+1) &= \mathbf{D}_b(k+1) - \mathbf{X}_P(k+1)\mathbf{W}_b(k+1) \\ &= \begin{bmatrix} \mathbf{X}_P(k+1) & \mathbf{D}_b(k+1) \end{bmatrix} \begin{bmatrix} -\mathbf{W}_b(k+1) \\ \mathbf{I} \end{bmatrix} \end{aligned} \tag{6.44}$$

where $\mathbf{D}_b(k+1)$ and $\mathbf{W}_b(k+1)$ are the respective desired response and coefficient vector matrices of the backward prediction problem.

Using the relation in (6.44), and assuming that the post-multiplication by permutation matrix $\mathbf{P}$ is already carried out, $\mathbf{X}_{P+M}(k+1)$ can be partitioned as

$$\mathbf{X}_{P+M}(k+1) = \left[\mathbf{X}_P(k+1)\ \mathbf{D}_b(k+1)\right]. \tag{6.45}$$

The process of obtaining lower triangular matrix $\mathbf{U}_{P+M}(k+1)$ from $\mathbf{X}_{P+M}(k+1)$ is performed as follows:

$$\begin{aligned}\mathbf{Q}_b(k+1)\mathbf{Q}(k)\mathbf{X}_{P+M}(k+1) &= \mathbf{Q}_b(k+1)\begin{bmatrix}\mathbf{0} & \mathbf{E}_{bq1}(k+1)\\ \mathbf{U}_P(k+1) & \mathbf{D}_{bq2}(k+1)\end{bmatrix}\\ &= \begin{bmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{E}_b(k+1)\\ \mathbf{U}_P(k+1) & \mathbf{D}_{bq2}(k+1)\end{bmatrix},\end{aligned} \tag{6.46}$$

where $\mathbf{Q}(k)$ contains $\mathbf{Q}_P(k+1)$ as a submatrix which triangulates $\mathbf{X}_P(k+1)$, generating $\mathbf{U}_P(k+1)$. Matrix $\mathbf{Q}_b(k+1)$ is responsible for generating the lower triangular matrix $\mathbf{E}_b(k+1)$ from $\mathbf{E}_{bq1}(k+1)$.

By removing the ever-increasing null section in (6.46), $\mathbf{U}_{P+M}(k+1)$ can be finally written as follows:

$$\mathbf{U}_{P+M}(k+1) = \begin{bmatrix}\mathbf{0} & \mathbf{E}_b(k+1)\\ \mathbf{U}_P(k+1) & \mathbf{D}_{bq2}(k+1)\end{bmatrix}. \tag{6.47}$$

The inverse of $\mathbf{U}_{P+M}(k+1)$ as given by (6.47) will be useful in further steps and is defined as

$$[\mathbf{U}_{P+M}(k+1)]^{-1} = \begin{bmatrix}-\mathbf{U}_P^{-1}(k+1)\mathbf{D}_{bq2}(k+1)\mathbf{E}_b^{-1}(k+1) & \mathbf{U}_P^{-1}(k+1)\\ \mathbf{E}_b^{-1}(k+1) & \mathbf{0}\end{bmatrix}. \tag{6.48}$$

Now, define **forward prediction** error matrix $\mathbf{E}^f(k+1)$ as

$$\begin{aligned}\mathbf{E}^f(k+1) = \mathbf{D}_f(k+1) - \begin{bmatrix}\mathbf{X}_P(k)\\ \mathbf{0}\end{bmatrix}\mathbf{W}_f(k+1) &= \begin{bmatrix}\mathbf{D}_f(k+1) & \mathbf{X}_P(k)\\ & \mathbf{0}^{\mathrm{T}}\end{bmatrix}\\ \times\begin{bmatrix}\mathbf{I}\\ \mathbf{W}_f(k+1)\end{bmatrix} &= \mathbf{X}_{P+M}(k+1)\begin{bmatrix}\mathbf{I}\\ \mathbf{W}_f(k+1)\end{bmatrix},\end{aligned} \tag{6.49}$$

where $\mathbf{D}_f(k+1)$ and $\mathbf{W}_f(k+1)$ are the desired response and the coefficient vector matrix of the backward prediction problem, respectively. A modified input data matrix $\bar{\mathbf{X}}_{P+M}(k+1)$ incorporating permutation matrix $\mathbf{P}$ is defined as follows.[2]

[2] Note that $\bar{\mathbf{X}}_{P+M}(k+1)$ is formed by adding $M-1$ rows of zeros to $\mathbf{X}_{P+M}(k+1)$ such that $\mathbf{U}_{P+M}(k+1)$ has the correct dimension in (6.55), i.e., $(P+M)\times(P+M)$.

$$\bar{\mathbf{X}}_{P+M}(k+1) = \begin{bmatrix} \mathbf{x}_{P+M}^{\mathrm{T}}(k+1) \\ \lambda^{1/2}\mathbf{x}_{P+M}^{\mathrm{T}}(k) \\ \vdots \\ \lambda^{(k+1)/2}\mathbf{x}_{P+M}^{\mathrm{T}}(0) \\ \mathbf{0}_{(M-1)\times(P+M)} \end{bmatrix} = \left[\begin{array}{c|c} \mathbf{D}_f(k+1) & \begin{matrix} \mathbf{X}_P(k) \\ \mathbf{0}^{\mathrm{T}} \end{matrix} \\ \hline \mathbf{0}_{(M-1)\times(P+M)} & \end{array}\right] \mathbf{P} \tag{6.50}$$

In order to triangulate $\bar{\mathbf{X}}_{P+M}(k+1)$ in (6.50) and obtain $\mathbf{U}_{P+M}(k+1)$, three sets of Givens rotation matrices $\mathbf{Q}(k)$, $\mathbf{Q}_f(k+1)$, and $\mathbf{Q}'_f(k+1)$ are needed [4, 5, 11]. The role of each matrix in the triangulation process is illustrated in the following equation.

$$\begin{aligned} &\mathbf{Q}'_f(k+1)\mathbf{Q}_f(k+1)\mathbf{Q}(k)\bar{\mathbf{X}}_{P+M}(k+1) = \mathbf{Q}'_f(k+1)\mathbf{Q}_f(k+1) \\ &\quad\times \begin{bmatrix} \mathbf{E}_{fq1}(k+1) & \mathbf{0} \\ \mathbf{D}_{fq2}(k+1) & \mathbf{U}_P(k) \\ \lambda^{(k+1)/2}\mathbf{x}_0^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0}_{(M-1)\times(P+M)} & \end{bmatrix} \mathbf{P} = \mathbf{Q}'_f(k+1) \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{D}_{fq2}(k+1) & \mathbf{U}_P(k) \\ \mathbf{E}_f(k+1) & \mathbf{0} \end{bmatrix} \mathbf{P} \end{aligned} \tag{6.51}$$

In (6.51), $\mathbf{Q}(k)$ contains $\mathbf{Q}_P(k)$ as a sub-matrix which triangulates $\mathbf{X}_P(k)$, generating $\mathbf{U}_P(k)$. Matrix $\mathbf{Q}_f(k+1)$ is responsible for the zeroing of matrix $\mathbf{E}_{fq1}(k+1)$. Note that, when working with fixed-order (or fixed-dimension, as opposed to the ever-increasing dimension of $\mathbf{Q}_P(k)$), this is equivalent to annihilating $\mathbf{e}_{fq1}^{\mathrm{T}}(k+1)$, the first row of $\mathbf{E}_{fq1}(k+1)$, against the diagonal of $\lambda^{1/2}\mathbf{E}_f(k)$, generating $\mathbf{E}_f(k+1)$, as shown in (6.54).

Removing the ever-increasing null section in (6.51) and using the fixed-order matrix $\mathbf{Q}'_{\theta f}(k+1)$ embedded in $\mathbf{Q}'_f(k+1)$, we obtain

$$\bar{\mathbf{U}}_{P+M}(k+1) = \mathbf{Q}'_{\theta f}(k+1) \begin{bmatrix} \mathbf{D}_{fq2}(k+1) & \mathbf{U}_P(k) \\ \mathbf{E}_f(k+1) & \mathbf{0} \end{bmatrix} \mathbf{P}. \tag{6.52}$$

Also starting from (6.51) and by using the fixed-order matrices $\mathbf{Q}_\theta(k)$ embedded in $\mathbf{Q}_P(k)$ and $\overline{\mathbf{Q}}_f(k+1)$ embedded in $\mathbf{Q}_f(k+1)$, we obtain after some algebraic manipulations the following equations:

$$\begin{bmatrix} \mathbf{e}_{fq1}^{\mathrm{T}}(k+1) \\ \mathbf{D}_{fq2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} \mathbf{x}_{k+1}^{\mathrm{T}} \\ \lambda^{1/2}\mathbf{D}_{fq2}(k) \end{bmatrix}, \tag{6.53}$$

where $\mathbf{x}_{k+1}^{\mathrm{T}} = [x_1(k+1) \quad x_2(k+1)\cdots x_M(k+1)]$ is the *forward reference signal* and $\mathbf{e}_{fq1}^{\mathrm{T}}(k+1)$ is the *rotated forward error*, and

$$\begin{bmatrix} \mathbf{0}^{\mathrm{T}} \\ \mathbf{E}_f(k+1) \end{bmatrix} = \overline{\mathbf{Q}}_f(k+1) \begin{bmatrix} \mathbf{e}_{fq1}^{\mathrm{T}}(k+1) \\ \lambda^{1/2}\mathbf{E}_f(k) \end{bmatrix}. \tag{6.54}$$

Let $\mathbf{E}_x^f(k+1) = [\mathbf{E}^f(k+1)]^{\mathrm{T}}\mathbf{E}^f(k+1)$ be the *forward prediction error covariance* matrix, where $\mathbf{E}^f(k+1)$ is the forward prediction error matrix defined in (6.49). Using $\bar{\mathbf{U}}_{P+M}(k+1)$ as defined in (6.52) instead of $\bar{\mathbf{X}}_{P+M}(k+1)$, it is straightforward to show that $\mathbf{E}_x^f(k+1) = \mathbf{E}_f^{\mathrm{T}}(k+1)\mathbf{E}_f(k+1)$. Thus, $M \times M$ lower triangular matrix $\mathbf{E}_f(k+1)$ in (6.52) and (6.54) can be interpreted as the Cholesky factor of the *forward prediction error covariance* matrix.

Note that the permutation matrix $\mathbf{P}$ in (6.52) prevents a direct annihilation of the first $M$ columns – corresponding to matrix $\mathbf{D}_{fq2}(k+1) = [\mathbf{d}_{fq2}^{(1)}(k+1)\ \mathbf{d}_{fq2}^{(2)}(k+1)\ \cdots\ \mathbf{d}_{fq2}^{(M)}(k+1)]$ – against the anti-diagonal of $\mathbf{E}_f(k+1)$ using the set of Givens rotations $\mathbf{Q}'_{\theta f}(k+1) = \mathbf{Q}'_{\theta f}{}^{(N)}(k+1)\cdots\mathbf{Q}'_{\theta f}{}^{(2)}(k+1)\mathbf{Q}'_{\theta f}{}^{(1)}(k+1)$. From (6.52) it can be seen that this permutation factor, $\mathbf{P} = \mathbf{P}_M\mathbf{P}_{M-1}\cdots\mathbf{P}_1$, will right-shift the first $M$ columns to position $p_i$, for $i = M$ to 1, in this order. Thus, only the first $P+i-p_i$ elements of each $\mathbf{d}_{fq2}^{(i)}(k+1)$ will be rotated against the anti-diagonal of $\mathbf{E}_f(k+1)$ using the set of Givens rotations in $\mathbf{Q}'_{\theta f}(k+1)$. It is straightforward to see that, when the position $p_i = i$, the corresponding permutation factor $\mathbf{P}_i$ degenerates to an identity matrix. If this is true for all $M$ channels, this formulation leads to the equal-order algorithms of [4–6, 11].

The process explained above is illustrated in Figure 6.4 (parts I–III) for a three-channel case with the first two channels having equal length, i.e., $p_1 = 1$ and $p_2 = 2$; consequently, $\mathbf{P}_1 = \mathbf{P}_2 = \mathbf{I}$. Part I of this figure shows the initial state as in (6.52) but with reduced dimension, and the operations involving matrices $\mathbf{Q}'_{\theta f}(k+1)$ and $\mathbf{P}$ are illustrated in parts II and III, respectively. As we can see, the resulting matrix $\bar{\mathbf{U}}_{P+M}(k+1)$ in (6.52) does not have the desired lower triangular shape, as depicted in part III of this figure. Hence, another permutation factor, $\overline{\mathbf{P}}$, is needed for up-shifting the $(P+M-i+1)$th row to the $(P+M-p_i+1)$th position (see Figure 6.4 – parts III–IV) leading to

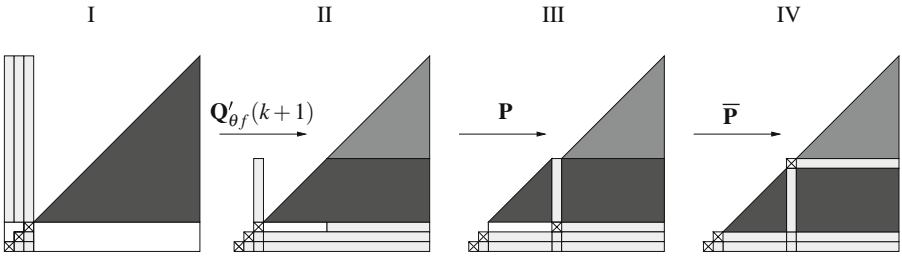


**Fig. 6.4** Obtaining the lower triangular $\mathbf{U}_{P+M}(k+1)$. The lighter color tone on top of parts II–IV denotes the matrix elements that have been rotated against the third line from the bottom by the third (and last) set of Givens rotations embedded in $\mathbf{Q}'_{\theta f}(k+1)$.

$$\mathbf{U}_{P+M}(k+1) = \overline{\mathbf{P}}\mathbf{Q}'_{\theta f}(k+1)\begin{bmatrix}\mathbf{D}_{fq2}(k+1) & \mathbf{U}_P(k)\\ \mathbf{E}_f(k+1) & \mathbf{0}\end{bmatrix}\mathbf{P}, \tag{6.55}$$

where permutation matrix $\overline{\mathbf{P}} = \mathbf{P}_1\mathbf{P}_2\cdots\mathbf{P}_M$.

From (6.55), it is possible to obtain

$$\begin{aligned}[\mathbf{U}_{P+M}(k+1)]^{-1} &= \mathbf{P}^{\mathrm{T}}\\ &\times\begin{bmatrix}\mathbf{0} & \mathbf{E}_f^{-1}(k+1)\\ \mathbf{U}_P^{-1}(k) & -\mathbf{U}_P^{-1}(k)\mathbf{D}_{fq2}(k+1)\mathbf{E}_f^{-1}(k+1)\end{bmatrix}\mathbf{Q}'^{\mathrm{T}}_{\theta f}(k+1)\overline{\mathbf{P}}^{\mathrm{T}},\end{aligned} \tag{6.56}$$

which will be used in the next section to derive the *a priori* and the *a posteriori* versions of the algorithm. Also from (6.55), we can write

$$\begin{bmatrix}\mathbf{0}\\ *\\ \mathbf{E}_f^0(k+1)\end{bmatrix} = \mathbf{Q}'_{\theta f}(k+1)\begin{bmatrix}\mathbf{D}_{fq2}(k+1)\\ \mathbf{E}_f(k+1)\end{bmatrix}, \tag{6.57}$$

where $\mathbf{E}_f^0(k+1)$ is the Cholesky factor of the zero-order error covariance matrix.[3] The *asterisk* $*$ denotes possible non-zero elements according to the process explained above.

### 6.4.2 A priori and A posteriori versions

If matrix $\mathbf{U}_{P+M}(k)$ is used to denote the Cholesky factor of $\mathbf{X}^{\mathrm{T}}_{P+M}(k)\mathbf{X}_{P+M}(k)$, we can define the *a priori* and *a posteriori* backward error vectors, $\mathbf{a}_{P+M}(k+1)$ and $\mathbf{f}_{P+M}(k+1)$, as follows:

$$\mathbf{a}_{P+M}(k+1) = \lambda^{-1/2}\mathbf{U}^{-T}_{P+M}(k)\mathbf{x}_{P+M}(k+1), \text{ and} \tag{6.58}$$

$$\mathbf{f}_{P+M}(k+1) = \mathbf{U}^{-T}_{P+M}(k+1)\mathbf{x}_{P+M}(k+1). \tag{6.59}$$

Vectors $\mathbf{a}_P(k+1)$ and $\mathbf{f}_P(k+1)$ are contained within matrix $\mathbf{Q}_\theta(k+1)$ [5, 11].

From (6.10), (6.56), and (6.58), we can write

$$\mathbf{a}_{P+M}(k+1) = \overline{\mathbf{P}}\lambda^{-1/2}\mathbf{Q}'_{\theta f}(k)\begin{bmatrix}\mathbf{a}_P(k)\\ \mathbf{r}(k+1)\end{bmatrix}, \tag{6.60}$$

[3] The term is coined due to the fact that, in the single-channel case, the corresponding scalar $\|\mathbf{e}_f^{(0)}(k+1)\| = \sum_{i=0}^{k+1}\lambda^{(k+1-i)}x^2(i)$ is the norm of the zero-order forward prediction error which is an estimate (albeit biased) of the input variance. Also note that, for zero-order prediction, the forward prediction error vector equals its backward counterpart, and $\|\mathbf{e}_f^{(0)}(k)\| = \|\mathbf{e}_b^{(0)}(k)\|$.

where

$$\begin{aligned}\mathbf{r}(k+1) &= \lambda^{-1/2}\mathbf{E}_f^{-T}(k)\left[\mathbf{x}_{k+1} - \mathbf{W}_f^T(k)\mathbf{x}_P(k)\right] \\ &= \lambda^{-1/2}\mathbf{E}_f^{-T}(k)\mathbf{e}'_f(k+1),\end{aligned} \tag{6.61}$$

with $\mathbf{e}'_f(k+1)$ being the *a priori* forward error vector. Thus, $\mathbf{r}(k+1)$ can be interpreted as the $N$th-order normalized *a priori* forward error vector. Matrix, $\mathbf{W}_f(k)$ given by

$$\mathbf{W}_f(k) = \mathbf{U}_P^{-1}(k-1)\mathbf{D}_{fq2}(k), \tag{6.62}$$

contains the coefficient vectors of the forward prediction problem. The matrix inversion operation in (6.61) can be avoided using the solution in [11], i.e.,

$$\begin{bmatrix} * \\ \mathbf{0} \end{bmatrix} = \overline{\mathbf{Q}}_f(k+1)\begin{bmatrix} 1/\gamma(k) \\ -\mathbf{r}(k+1) \end{bmatrix}. \tag{6.63}$$

Combining (6.48), (6.58), and the definition of the input vector in (6.10), $\mathbf{a}_{P+M}(k+1)$ can be expressed as

$$\mathbf{a}_{P+M}(k+1) = \begin{bmatrix} \mathbf{a}^{(N)}(k+1) \\ \mathbf{a}_P(k+1) \end{bmatrix}, \tag{6.64}$$

where the $M \times 1$ element vector of $\mathbf{a}_{P+M}(k+1)$, $\mathbf{a}^{(N)}(k+1)$, is given by

$$\begin{aligned}\mathbf{a}^{(N)}(k+1) &= \lambda^{-1/2}\mathbf{E}_b^{-T}(k)\left[\mathbf{x}_{k-N+1} - \mathbf{W}_b^T(k)\mathbf{x}_P(k+1)\right] \\ &= \lambda^{-1/2}\mathbf{E}_b^{-T}(k)\mathbf{e}'_b(k+1),\end{aligned} \tag{6.65}$$

with $\mathbf{e}'_b(k+1)$ being the $N$th-order *a priori* backward error vector and matrix $\mathbf{W}_b(k)$ contains the coefficient vectors of the backward prediction problem. From the last equation, $\mathbf{a}^{(N)}(k+1)$ can be thought as the $N$th-order normalized *a priori* backward error vector. [4]

Using similar procedure, combining Equations (6.10), (6.56), and (6.59), yields

$$\mathbf{f}_{P+M}(k+1) = \overline{\mathbf{P}}\mathbf{Q}'_{\theta f}(k+1)\begin{bmatrix} \mathbf{f}_P(k) \\ \mathbf{p}(k+1) \end{bmatrix}, \tag{6.66}$$

where

$$\begin{aligned}\mathbf{p}(k+1) &= \mathbf{E}_f^{-T}(k+1)\left[\mathbf{x}_{k+1} - \mathbf{W}_f^T(k+1)\mathbf{x}_P(k)\right] \\ &= \mathbf{E}_f^{-T}(k+1)\mathbf{e}_f(k+1),\end{aligned} \tag{6.67}$$

[4] As shall be seen in Section 6.5, this argument can be taken further to demonstrate that $\mathbf{a}_{P+M}(k+1)$ is actually a nesting of vectors $\mathbf{a}^{(j)}(k+1)$ of size $M \times 1$, for $j = 0, 1, .., N$. Similar observation holds for vector $\mathbf{f}_{P+M}(k+1)$.

with $\mathbf{e}_f(k+1)$ being the *a posteriori* forward error vector. Therefore, $\mathbf{p}(k+1)$ is interpreted as the $N$th-order normalized *a posteriori* forward error vector. Matrix $\mathbf{W}_f(k+1)$ contains the coefficient vectors of the forward prediction problem.

Now, from (6.48), (6.59), and the definition of the input vector in (6.10), $\mathbf{f}_{P+M}(k+1)$ can be partitioned as

$$\mathbf{f}_{P+M}(k+1) = \begin{bmatrix} \mathbf{f}^{(N)}(k+1) \\ \mathbf{f}_P(k+1) \end{bmatrix}, \tag{6.68}$$

where vector $\mathbf{f}^{(N)}(k+1)$ is given by

$$\begin{aligned} \mathbf{f}^{(N)}(k+1) &= \mathbf{E}_b^{-T}(k+1)\left[\mathbf{x}_{k-N+1} - \mathbf{W}_b^T(k+1)\mathbf{x}_P(k+1)\right] \\ &= \mathbf{E}_b^{-T}(k+1)\mathbf{e}_b(k+1), \end{aligned} \tag{6.69}$$

with $\mathbf{e}_b(k+1)$ being the $N$th-order *a posteriori* backward error vector and matrix $\mathbf{W}_b(k+1)$ contains the coefficient vectors of the backward prediction problem. Hence, $\mathbf{f}^{(N)}(k+1)$ can be regarded as the $N$th-order normalized *a posteriori* backward error vector.

To solve for $\mathbf{p}(k+1)$ avoiding the matrix inversion in (6.67), we can use

$$\overline{\mathbf{Q}}_f(k+1)\begin{bmatrix} \gamma(k) \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} * \\ \mathbf{p}(k+1) \end{bmatrix}. \tag{6.70}$$

*Proof.* From (6.54), it is clear that $\mathbf{E}_f(k+1)$ is the Cholesky factor of

$$[\tilde{\mathbf{e}}_{fq1} \quad \lambda^{1/2}\mathbf{E}_f^{\mathrm{T}}(k)][\tilde{\mathbf{e}}_{fq1} \quad \lambda^{1/2}\mathbf{E}_f^{\mathrm{T}}(k)]^{\mathrm{T}}. \tag{6.71}$$

Hence, (6.54) can be written in a product form as follows [14]:

$$\mathbf{E}_f^{\mathrm{T}}(k+1)\mathbf{E}_f(k+1) = \tilde{\mathbf{e}}_{fq1}(k+1)\tilde{\mathbf{e}}_{fq1}^{\mathrm{T}}(k+1) + \lambda\mathbf{E}_f^{\mathrm{T}}(k)\mathbf{E}_f(k). \tag{6.72}$$

Pre-multiply and post-multiply (6.72) by $\mathbf{E}_f^{-T}(k+1)\gamma^2(k)$ and $\mathbf{E}_f^{-1}(k+1)$, respectively. After some algebraic manipulations, we have

$$\gamma^2(k)\mathbf{I} = \mathbf{p}(k+1)\mathbf{p}^{\mathrm{T}}(k+1) + \boldsymbol{\Psi} \tag{6.73}$$

where $\boldsymbol{\Psi} = \lambda\gamma^2(k)\mathbf{E}_f^{-T}(k+1)\mathbf{E}_f^T(k)\mathbf{E}_f(k)\mathbf{E}_f^{-1}(k+1)$.

Finally, after pre-multiplying and post-multiplying (6.73) by $\mathbf{p}^{\mathrm{T}}(k+1)$ and $\mathbf{p}(k+1)$, respectively, and dividing the result by $\mathbf{p}^{\mathrm{T}}(k+1)\mathbf{p}(k+1)$, we obtain

$$\begin{aligned} \gamma^2(k) &= \mathbf{p}^{\mathrm{T}}(k+1)\mathbf{p}(k+1) + \frac{\mathbf{p}^{\mathrm{T}}(k+1)\boldsymbol{\Psi}\mathbf{p}(k+1)}{\mathbf{p}^{\mathrm{T}}(k+1)\mathbf{p}(k+1)} \\ &= \mathbf{p}^{\mathrm{T}}(k+1)\mathbf{p}(k+1) + *^2. \end{aligned} \tag{6.74}$$

The expression in (6.74) can be regarded as a Cholesky product. Hence, it can be factored as

$$\begin{bmatrix} \gamma(k) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q} \begin{bmatrix} * \\ \mathbf{p}(k+1) \end{bmatrix}, \tag{6.75}$$

where $\mathbf{Q}$ is an orthogonal matrix.

If we recall our starting point in (6.54), we can see that $\mathbf{Q}$ is related to $\overline{\mathbf{Q}}_f(k+1)$. Moreover, from the knowledge of the internal structure of $\overline{\mathbf{Q}}_f(k+1)$, we can conclude that $\mathbf{Q} = \overline{\mathbf{Q}}_f^{\mathrm{T}}(k+1)$ satisfies (6.75) leading to (6.70). Vector $\mathbf{p}(k+1)$ can be easily obtained from (6.70) because $\gamma(k)$ and $\overline{\mathbf{Q}}_f(k+1)$ are known quantities. The reader can use similar arguments in order to prove (6.63).

The rotation angles in matrix $\mathbf{Q}_\theta(k)$ are obtained using

$$\mathbf{Q}_\theta(k+1) \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \gamma(k+1) \\ \mathbf{f}_P(k+1) \end{bmatrix}, \tag{6.76}$$

for the *a posteriori* case, and

$$\begin{bmatrix} 1/\gamma(k+1) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} 1 \\ -\mathbf{a}_P(k+1) \end{bmatrix}, \tag{6.77}$$

for the *a priori* case.

Finally, the joint process estimation is performed as

$$\begin{bmatrix} e_{q1}(k+1) \\ \mathbf{d}_{q2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\mathbf{d}_{q2}(k) \end{bmatrix}, \tag{6.78}$$

and the *a priori error* is given by [5, 11]

$$e(k+1) = e_{q1}(k+1)/\gamma(k+1). \tag{6.79}$$

The *a posteriori* and *a priori* block-MC-FQRD-RLS algorithms are summarized in Tables 6.4 and 6.5, respectively. In these tables, the common equations are in gray in order to highlight the difference between both algorithms.

### *6.4.3 Alternative implementations*

Similar to their sequential-channel processing counterparts, alternative implementations for the block-channel algorithms are available when the $M$ channels are of equal order, i.e., $N_i = N$ and $P = MN$. In this particular case, the last $M$ elements of vectors $\mathbf{a}_{P+M}(k+1)$ and $\mathbf{f}_{P+M}(k+1)$ are known prior to their updating through Equations (6.58) and (6.59), respectively [6].

**Table 6.4** Equations of the *a posteriori* block-MCFQRD based on the update of backward prediction errors [5–7].

| MCFQRD_POS_B |
|---|
| For each $k$, do |
| { 1. Obtaining $\mathbf{D}_{fq2}(k+1)$ and $\mathbf{e}_{fq1}(k+1)$ |
| $\begin{bmatrix} \mathbf{e}^{\mathrm{T}}_{fq1}(k+1) \\ \mathbf{D}_{fq2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} \mathbf{x}^{\mathrm{T}}_{k+1} \\ \lambda^{1/2}\mathbf{D}_{fq2}(k) \end{bmatrix}$ (6.53) |
| 2. Obtaining $\mathbf{E}_f(k+1)$ and $\overline{\mathbf{Q}}_f(k+1)$ |
| $\begin{bmatrix} \mathbf{0}^{\mathrm{T}} \\ \mathbf{E}_f(k+1) \end{bmatrix} = \overline{\mathbf{Q}}_f(k+1) \begin{bmatrix} \mathbf{e}^{\mathrm{T}}_{fq1}(k+1) \\ \lambda^{1/2}\mathbf{E}_f(k) \end{bmatrix}$ (6.54) |
| 3. Obtaining $\mathbf{p}(k+1)$ |
| $\begin{bmatrix} * \\ \mathbf{p}(k+1) \end{bmatrix} = \overline{\mathbf{Q}}_f(k+1) \begin{bmatrix} \gamma(k) \\ \mathbf{0} \end{bmatrix}$ implements (6.67) |
| 4. Obtaining $\mathbf{Q}'_{\theta f}(k+1)$ |
| $\begin{bmatrix} \mathbf{0} \\ * \\ \mathbf{E}^0_f(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}(k+1) \begin{bmatrix} \mathbf{D}_{fq2}(k+1) \\ \mathbf{E}_f(k+1) \end{bmatrix}$ (6.57) |
| 5. Obtaining $\mathbf{f}_P(k+1)$ |
| $\mathbf{f}_{P+M}(k+1) = \overline{\mathbf{P}}\mathbf{Q}'_{\theta f}(k+1) \begin{bmatrix} \mathbf{f}_P(k) \\ \mathbf{p}(k+1) \end{bmatrix}$ (6.66) |
| 6. Obtaining $\mathbf{Q}_\theta(k+1)$ and $\gamma(k+1)$ |
| $\mathbf{Q}_\theta(k+1) \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \gamma(k+1) \\ \mathbf{f}_P(k+1) \end{bmatrix}$ (6.76) |
| 7. Joint estimation |
| $\begin{bmatrix} e_{q1}(k+1) \\ \mathbf{d}_{q2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\mathbf{d}_{q2}(k) \end{bmatrix}$ (6.78) |
| 8. Obtaining the *a priori* error |
| $e(k+1) = e_{q1}(k+1)/\gamma(k+1)$ (6.79) |
| } |

Assuming $\overline{\mathbf{P}} = \mathbf{P} = \mathbf{I}$, after the multiplication by $\mathbf{Q}'_{\theta f}(k+1)$ is carried out in (6.55), matrix $\mathbf{U}_{P+M}(k+1)$ can be partitioned as

$$\mathbf{U}_{P+M}(k+1) = \begin{bmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{E}^0_f(k+1) & \mathbf{C} \end{bmatrix} \tag{6.80}$$

and by taking its inverse, yields

$$[\mathbf{U}_{P+M}(k+1)]^{-1} = \begin{bmatrix} -\left[\mathbf{E}^0_f(k+1)\right]^{-1}\mathbf{C}\mathbf{B}^{-1} & \left[\mathbf{E}^0_f(k+1)\right]^{-1} \\ \mathbf{B}^{-1} & \mathbf{0} \end{bmatrix}. \tag{6.81}$$

If we now use (6.81) together with (6.58), (6.59), and (6.10), vectors $\mathbf{a}_{P+M}(k+1)$ and $\mathbf{f}_{P+M}(k+1)$ can be written, respectively, as

$$\mathbf{a}_{P+M}(k+1) = \begin{bmatrix} * \\ \lambda^{-1/2}\left[\mathbf{E}^0_f(k)\right]^{-T}\mathbf{x}_{k+1} \end{bmatrix}, \text{ and} \tag{6.82}$$

**Table 6.5** Equations of the *a priori* block-MCFQRD based on the update of backward prediction errors [6, 7, 11].

MCFQRD_PRI_B

For each $k$, do

{ 1. Obtaining $\mathbf{D}_{fq2}(k+1)$ and $\mathbf{e}_{fq1}(k+1)$

$$\begin{bmatrix} \mathbf{e}^{\mathrm{T}}_{fq1}(k+1) \\ \mathbf{D}_{fq2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} \mathbf{x}^{\mathrm{T}}_{k+1} \\ \lambda^{1/2}\mathbf{D}_{fq2}(k) \end{bmatrix} \tag{6.53}$$

2. Obtaining $\mathbf{E}_f(k+1)$ and $\overline{\mathbf{Q}}_f(k+1)$

$$\begin{bmatrix} \mathbf{0}^{\mathrm{T}} \\ \mathbf{E}_f(k+1) \end{bmatrix} = \overline{\mathbf{Q}}_f(k+1) \begin{bmatrix} \mathbf{e}^{\mathrm{T}}_{fq1}(k+1) \\ \lambda^{1/2}\mathbf{E}_f(k) \end{bmatrix} \tag{6.54}$$

3. Obtaining $\mathbf{r}(k+1)$

$$\begin{bmatrix} * \\ \mathbf{0} \end{bmatrix} = \overline{\mathbf{Q}}_f(k+1) \begin{bmatrix} 1/\gamma(k) \\ -\mathbf{r}(k+1) \end{bmatrix} \quad \text{implements (6.61)}$$

4. Obtaining $\mathbf{a}_P(k+1)$

$$\mathbf{a}_{P+M}(k+1) = \overline{\mathbf{P}}\mathbf{Q}'_{\theta f}(k) \begin{bmatrix} \mathbf{a}_P(k) \\ \mathbf{r}(k+1) \end{bmatrix} \tag{6.60}$$

5. Obtaining $\mathbf{Q}'_{\theta f}(k+1)$

$$\begin{bmatrix} \mathbf{0} \\ * \\ \mathbf{E}^0_f(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}(k+1) \begin{bmatrix} \mathbf{D}_{fq2}(k+1) \\ \mathbf{E}_f(k+1) \end{bmatrix} \tag{6.57}$$

6. Obtaining $\mathbf{Q}_\theta(k+1)$ and $\gamma(k+1)$

$$\begin{bmatrix} 1/\gamma(k+1) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} 1 \\ -\mathbf{a}_P(k+1) \end{bmatrix} \tag{6.77}$$

7. Joint estimation

$$\begin{bmatrix} e_{q1}(k+1) \\ \mathbf{d}_{q2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\mathbf{d}_{q2}(k) \end{bmatrix} \tag{6.78}$$

8. Obtaining the *a priori* error

$$e(k+1) = e_{q1}(k+1)/\gamma(k+1) \tag{6.79}$$

}

$$\mathbf{f}_{P+M}(k+1) = \begin{bmatrix} * \\ \left[\mathbf{E}^0_f(k+1)\right]^{-T}\mathbf{x}_{k+1} \end{bmatrix}. \tag{6.83}$$

From (6.82) and (6.83), we can see that the last $M$ elements of $\mathbf{a}_{P+M}(k+1)$ and $\mathbf{f}_{P+M}(k+1)$ are known quantities. The alternative implementations of these algorithms arise when the recursive updating of these vectors is performed based on this *a priori* knowledge.

## 6.5 Order-Recursive MC-FQRD-RLS Algorithms

Block multichannel algorithms are more suitable for order-recursive implementations and parallel processing as compared to their sequential-channel counterparts. Therefore, only the formers shall be addressed in this section. Sequential-channel processing algorithms can also be implemented order-recursively up to a certain

degree [9, 11], however, adding order-recursiveness to the already existing channel recursive nature, leads to more complicated structures.

For sake of simplicity, the special case of all $M$ channels having equal orders, i.e., $N_i = N$, is considered. We shall start by revisiting the definitions of Cholesky factor of the information matrix, $\mathbf{U}_{P+M}(k+1)$, given by Equations (6.47) and (6.55), obtained from solving the backward and the forward prediction problems, respectively, and reproduced below assuming channels of equal orders.

$$\mathbf{U}_{P+M}(k+1) = \mathbf{U}_{N+1}(k+1) = \begin{bmatrix} \mathbf{0} & \mathbf{E}_b^N(k+1) \\ \mathbf{U}_N(k+1) & \mathbf{D}_{bq2}^N(k+1) \end{bmatrix} \tag{6.84}$$

$$= \mathbf{Q}'_{\theta f}(k+1) \begin{bmatrix} \mathbf{D}_{fq2}^N(k+1) & \mathbf{U}_N(k) \\ \mathbf{E}_f^N(k+1) & \mathbf{0} \end{bmatrix} \tag{6.85}$$

The superscript $N$ added to variables $\mathbf{E}_b(k+1)$, $\mathbf{D}_{bq2}$, $\mathbf{D}_{fq2}(k+1)$, and $\mathbf{E}_f(k+1)$ emphasizes that these quantities are related to the $N$th-order prediction problem.[5] The last two equations can be written in a generalized form as

$$\mathbf{U}_{j+1}(k+1) = \begin{bmatrix} \mathbf{0} & \mathbf{E}_b^{(j)}(k+1) \\ \mathbf{U}_j(k+1) & \mathbf{D}_{bq2}^{(j)}(k+1) \end{bmatrix} \tag{6.86}$$

$$= \mathbf{Q}'^{(j)}_{\theta f}(k+1) \begin{bmatrix} \mathbf{D}_{fq2}^{(j)}(k+1) & \mathbf{U}_j(k) \\ \mathbf{E}_f^{(j)}(k+1) & \mathbf{0} \end{bmatrix}, \tag{6.87}$$

for $j = 0, 1, \ldots, N$. This property is the key to derive the order-recursive versions of the algorithms. Indeed, the information provided by (6.86) and (6.87) justifies the generalization of Equations (6.65) and (6.69) from Section 6.4.2, respectively, as

$$\mathbf{a}^{(j)}(k+1) = \lambda^{-1/2} \left[\mathbf{E}_b^{(j)}(k)\right]^{-T} \mathbf{e}_b'^{(j)}(k+1) \tag{6.88}$$

and

$$\mathbf{f}^{(j)}(k+1) = \left[\mathbf{E}_b^{(j)}(k+1)\right]^{-T} \mathbf{e}_b^{(j)}(k+1) \tag{6.89}$$

where $\mathbf{e}_b'^{(j)}(k+1)$ and $\mathbf{e}_b^{(j)}(k+1)$ are, respectively, the $j$th-order *a priori* and *a posteriori* backward error vectors, for $j = 0, 1, \ldots, N$. Therefore, vectors $\mathbf{a}_{P+M}(k+1)$ and $\mathbf{f}_{P+M}(k+1)$ can be regarded as a nesting of $N+1$ subvectors of size $M \times 1$, i.e.,

$$\mathbf{a}_{N+1}(k+1) = \begin{bmatrix} \mathbf{a}^{(N)}(k+1) \\ \mathbf{a}^{(N-1)}(k+1) \\ \vdots \\ \mathbf{a}^{(0)}(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{a}^{(N)}(k+1) \\ \mathbf{a}_N(k+1) \end{bmatrix} \tag{6.90}$$

[5] Note that the subscripts $_{P+M}$ and $_{N+1}$ are interchangeable and $_{N+1}$ is used whenever the order of the prediction problems needs to be highlighted, whereas $_{P+M}$ emphasizes vectors or matrices dimensions.

and

$$\mathbf{f}_{N+1}(k+1) = \begin{bmatrix} \mathbf{f}^{(N)}(k+1) \\ \mathbf{f}^{(N-1)}(k+1) \\ \vdots \\ \mathbf{f}^{(0)}(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{f}^{(N)}(k+1) \\ \mathbf{f}_N(k+1) \end{bmatrix}. \tag{6.91}$$

Recalling that $\mathbf{Q}'_{\theta f}(k)$ and $\mathbf{Q}'_{\theta f}(k+1)$ are used to update $\mathbf{a}_{N+1}(k)$ and $\mathbf{f}_{N+1}(k)$, respectively, we can finally rewrite Equations (6.60) and (6.66) into an order-recursive form, i.e., for $j = 1, 2, \ldots, N$, as

$$\begin{bmatrix} \mathbf{0}_{M(N-j)} \\ \mathbf{a}^{(j)}(k+1) \\ \mathbf{0}_{M(j-1)} \\ \mathbf{r}_{j-1}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}{}^{(j)}(k) \begin{bmatrix} \mathbf{0}_{M(N-j)} \\ \mathbf{a}^{(j-1)}(k) \\ \mathbf{0}_{M(j-1)} \\ \mathbf{r}_j(k+1) \end{bmatrix}, \tag{6.92}$$

where $\mathbf{r}_j(k+1)$ is the $j$th-order normalized *a priori* forward error vector, and

$$\begin{bmatrix} \mathbf{0}_{M(N-j)} \\ \mathbf{f}^{(j)}(k+1) \\ \mathbf{0}_{M(j-1)} \\ \mathbf{p}_{j-1}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}{}^{(j)}(k+1) \begin{bmatrix} \mathbf{0}_{M(N-j)} \\ \mathbf{f}^{(j-1)}(k) \\ \mathbf{0}_{M(j-1)} \\ \mathbf{p}_j(k+1) \end{bmatrix} \tag{6.93}$$

where $\mathbf{p}_j(k+1)$ is the $j$th-order normalized *a posteriori* forward error vector.

Equation (6.93) implements the order-recursive counterpart of step 5 (Table 6.4) of the *a posteriori* version of the MC-FQRD-RLS algorithm, whereas (6.92) stands for corresponding order-recursive implementation of step 4 (Table 6.5) of the *a priori* version.

Now we shall see how to compute the set of Givens rotations in $\mathbf{Q}'_{\theta f}{}^{(j)}(k+1)$. Specifically, we shall find a way to carry out steps 4 and 5 of algorithms in Tables 6.4 and 6.5, respectively, in an order-recursive manner. We begin with noting that previous arguments support the partitioning of matrix $\mathbf{D}_{fq2}(k+1)$ into $N$ $M \times M$-blocks as follows:

$$\mathbf{D}_{fq2}(k+1) = \begin{bmatrix} \mathbf{D}^{(1)}_{fq2}(k+1) \\ \vdots \\ \mathbf{D}^{(N)}_{fq2}(k+1) \end{bmatrix}. \tag{6.94}$$

Now, recalling (6.87), we realize that (6.57) can be rewritten as

$$\begin{bmatrix} \mathbf{0}_{M(N-j+1)\times M} \\ \mathbf{0}_{M(j-1)\times M} \\ \mathbf{E}_f^{(j-1)}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}{}^{(j)}(k+1) \begin{bmatrix} \mathbf{0}_{M(j-1)\times M} \\ \mathbf{D}^{(j)}_{fq2}(k+1) \\ \mathbf{0}_{M(N-j)\times M} \\ \mathbf{E}_f^{(j)}(k+1) \end{bmatrix} \tag{6.95}$$

for $j = 1,2,\ldots,N$. Moreover, the inherent order-recursiveness of previous equation leads us to conclude that matrix $\mathbf{Q}'_{\theta f}(k+1)$ in (6.57) can be regarded as a product of the form:

$$\mathbf{Q}'_{\theta f}(k+1) = \mathbf{Q}'_{\theta f}{}^{(N)}(k+1)\mathbf{Q}'_{\theta f}{}^{(N-1)}(k+1)\cdots\mathbf{Q}'_{\theta f}{}^{(1)}(k+1), \quad (6.96)$$

where each $\mathbf{Q}'_{\theta f}{}^{(j)}(k+1)$, for $j = 1,2,\ldots,N$, is a product itself of $M^2$ elementary Givens rotation matrices.

As for step 6 (Tables 6.4 and 6.5), it is straightforward to see that the rotation angles $\mathbf{Q}_{\theta}^{(j)}(k+1)$ are now obtained through

**Table 6.6** Algorithm number **1** of Table 6.1 [4].

**Lattice block-MCFQRD_POS_B**

*Initializations:*

$\mathbf{f}_P(0) = 0$; $\mathbf{D}_{fq2}(0) = 0$; $\gamma_0(0) = 1$; $\mathbf{d}_{q2}(0) = 0$; $\mathbf{E}_f^j(0) = \mu\mathbf{I}$,
$\mu = small\ \ number$, all $cosines = 1$, and all $sines = 0$;
For each $k$, do
{ $\widetilde{\mathbf{e}}_{fq1}^{(0)\,T}(k+1) = \mathbf{x}_{k+1}^{\mathrm{T}}$;
Obtaining $\mathbf{E}_f^{(0)}(k+1)$ and $\mathbf{p}_0(k+1)$:

$$\begin{bmatrix} \mathbf{0}^{\mathrm{T}} & * \\ \mathbf{E}_f^{(0)}(k+1) & \mathbf{p}_0(k+1) \end{bmatrix} = \overline{\mathbf{Q}}_f^{(0)}(k+1) \begin{bmatrix} \widetilde{\mathbf{e}}_{fq1}^{(0)\,T}(k+1) & \gamma_0(k) \\ \lambda^{1/2}\mathbf{E}_f^{(0)}(k) & \mathbf{0} \end{bmatrix};$$

$\mathbf{f}^{(N+1)}(k+1) = \mathbf{p}_0(k+1)$; $\gamma_0(k+1) = 1$;
$\mathbf{e}_{q1}(k+1) = d(k+1)$;
for $j = 1:N$
{ 1. Obtaining $\mathbf{D}_{fq2}^{(j)}(k+1)$ and $\mathbf{e}_{fq1}^{(j)}(k+1)$:

$$\begin{bmatrix} \widetilde{\mathbf{e}}_{fq1}^{(j)\,\mathrm{T}}(k+1) \\ \mathbf{D}_{fq2}^{(j)}(k+1) \end{bmatrix} = \mathbf{Q}_{\theta}^{(j)}(k) \begin{bmatrix} \widetilde{\mathbf{e}}_{fq1}^{(j-1)\,\mathrm{T}}(k+1) \\ \lambda^{1/2}\mathbf{D}_{fq2}^{(j)}(k) \end{bmatrix};$$

2. Obtaining $\mathbf{E}_f^{(j)}(k+1)$ and $\mathbf{p}_j(k+1)$:

$$\begin{bmatrix} \mathbf{0}^{\mathrm{T}} & * \\ \mathbf{E}_f^{(j)}(k+1) & \mathbf{p}_j(k+1) \end{bmatrix} = \overline{\mathbf{Q}}_f^{(j)}(k+1) \begin{bmatrix} \widetilde{\mathbf{e}}_{fq1}^{(j)\,\mathrm{T}}(k+1) & \gamma_j(k) \\ \lambda^{1/2}\mathbf{E}_f^{(j)}(k) & \mathbf{0} \end{bmatrix};$$

3. Obtaining $\mathbf{Q}'_{\theta f}{}^{(j)}(k+1)$:

$$\begin{bmatrix} \mathbf{0}_{M(N-j+1)\times M} \\ \mathbf{0}_{M(j-1)\times M} \\ \mathbf{E}_f^{(j-1)}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}{}^{(j)}(k+1) \begin{bmatrix} \mathbf{0}_{M(j-1)\times M} \\ \mathbf{D}_{fq2}^{(j)}(k+1) \\ \mathbf{0}_{M(N-j)\times M} \\ \mathbf{E}_f^{(j)}(k+1) \end{bmatrix};$$

4. Obtaining $\mathbf{f}^{(j)}(k+1)$:

$$\begin{bmatrix} \mathbf{0}_{M(N-j)} \\ \mathbf{f}^{(j)}(k+1) \\ \mathbf{0}_{M(j-1)} \\ \mathbf{p}_{j-1}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}{}^{(j)}(k+1) \begin{bmatrix} \mathbf{0}_{M(N-j)} \\ \mathbf{f}^{(j-1)}(k) \\ \mathbf{0}_{M(j-1)} \\ \mathbf{p}_j(k+1) \end{bmatrix};$$

5. Obtaining $\mathbf{Q}_{\theta}^{(j)}(k+1)$ and $\gamma_j(k+1)$:

$$\mathbf{Q}_{\theta}^{(j)}(k+1) \begin{bmatrix} \gamma_{j-1}(k+1) \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \gamma_j(k+1) \\ \mathbf{f}^{(j-1)}(k+1) \end{bmatrix};$$

6. Joint estimation:

$$\begin{bmatrix} \mathbf{e}_{q1}^{(j)}(k+1) \\ \mathbf{d}_{q2}^{(j)}(k+1) \end{bmatrix} = \mathbf{Q}_{\theta}^{(j)}(k+1) \begin{bmatrix} \mathbf{e}_{q1}^{(j-1)}(k+1) \\ \lambda^{1/2}\mathbf{d}_{q2}^{(j)}(k) \end{bmatrix};$$

7. Obtaining the *a priori* error:
$e_j(k+1) = e_{q1}^{(j)}(k+1)/\gamma_j(k+1)$;
} % for $j$
} % for $k$

$$\begin{bmatrix} 1/\gamma_j(k+1) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta^{(j)}(k+1) \begin{bmatrix} 1/\gamma_{j-1}(k+1) \\ -\mathbf{a}^{(j-1)}(k+1) \end{bmatrix} \tag{6.97}$$

for the *a priori* algorithm, and

$$\mathbf{Q}_\theta^{(j)}(k+1) \begin{bmatrix} \gamma_{j-1}(k+1) \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \gamma_j(k+1) \\ \mathbf{f}^{(j-1)}(k+1) \end{bmatrix} \tag{6.98}$$

for the *a posteriori* case. The joint estimation (step 7) is performed according to

$$\begin{bmatrix} \mathbf{e}_{q1}^{(j)}(k+1) \\ \mathbf{d}_{q2}^{(j)}(k+1) \end{bmatrix} = \mathbf{Q}_\theta^{(j)}(k+1) \begin{bmatrix} \mathbf{e}_{q1}^{(j-1)}(k+1) \\ \lambda^{1/2}\mathbf{d}_{q2}^{(j)}(k) \end{bmatrix}. \tag{6.99}$$

**Table 6.7** Algorithm number **9** of Table 6.1 [11].

**Lattice block-MCFQR_PRI_B**

*Initializations:*

$\mathbf{a}_P(0) = 0$; $\mathbf{D}_{fq2}(0) = 0$; $\gamma_0(0) = 1$; $\mathbf{d}_{q2}(0) = 0$; $\mathbf{E}_f^j(0) = \mu\mathbf{I}$,

$\mu = small\ number$, all $cosines = 1$, and all $sines = 0$;

For each $k$, do

{ $\widetilde{\mathbf{e}}_{fq1}^{(0)\,T}(k+1) = \mathbf{x}_{k+1}^{\mathrm{T}}$;

Obtaining $\mathbf{E}_f^{(0)}(k+1)$ and $\mathbf{r}_0(k+1)$:

$$\begin{bmatrix} \mathbf{0}^{\mathrm{T}} & * \\ \mathbf{E}_f^{(0)}(k+1) & \mathbf{0} \end{bmatrix} = \overline{\mathbf{Q}}_f^{(0)}(k+1) \begin{bmatrix} \widetilde{\mathbf{e}}_{fq1}^{(0)\,T}(k+1) & 1/\gamma_0(k) \\ \lambda^{1/2}\mathbf{E}_f^{(0)}(k) & -\mathbf{r}_0(k+1) \end{bmatrix};$$

$\mathbf{a}^{(0)}(k+1) = \mathbf{r}_0(k+1)$; $\gamma_0(k+1) = 1$;

$\mathbf{e}_{q1}(k+1) = d(k+1)$;

for $j = 1 : N$

{ 1. Obtaining $\mathbf{D}_{fq2}^{(j)}(k+1)$ and $\mathbf{e}_{fq1}^{(j)}(k+1)$:

$$\begin{bmatrix} \widetilde{\mathbf{e}}_{fq1}^{(j)\,\mathrm{T}}(k+1) \\ \mathbf{D}_{fq2}^{(j)}(k+1) \end{bmatrix} = \mathbf{Q}_\theta^{(j)}(k) \begin{bmatrix} \widetilde{\mathbf{e}}_{fq1}^{(j-1)\,\mathrm{T}}(k+1) \\ \lambda^{1/2}\mathbf{D}_{fq2}^{(j)}(k) \end{bmatrix};$$

2. Obtaining $\mathbf{E}_f^{(j)}(k+1)$ and $\mathbf{p}_j(k+1)$:

$$\begin{bmatrix} \mathbf{0}^{\mathrm{T}} & * \\ \mathbf{E}_f^{(j)}(k+1) & \mathbf{0} \end{bmatrix} = \overline{\mathbf{Q}}_f^{(j)}(k+1) \begin{bmatrix} \widetilde{\mathbf{e}}_{fq1}^{(j)\,\mathrm{T}}(k+1) & 1/\gamma_j(k) \\ \lambda^{1/2}\mathbf{E}_f^{(j)}(k) & -\mathbf{r}_j(k+1) \end{bmatrix};$$

3. Obtaining $\mathbf{a}^{(j)}(k+1)$:

$$\begin{bmatrix} \mathbf{0}_{M(N-j)} \\ \mathbf{a}^{(j)}(k+1) \\ \mathbf{0}_{M(j-1)} \\ \mathbf{r}_{j-1}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}{}^{(j)}(k) \begin{bmatrix} \mathbf{0}_{M(N-j)} \\ \mathbf{a}^{(j-1)}(k) \\ \mathbf{0}_{M(j-1)} \\ \mathbf{r}_j(k+1) \end{bmatrix};$$

4. Obtaining $\mathbf{Q}'_{\theta f}{}^{(j)}(k+1)$:

$$\begin{bmatrix} \mathbf{0}_{M(N-j+1)\times M} \\ \mathbf{0}_{M(j-1)\times M} \\ \mathbf{E}_f^{(j-1)}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}{}^{(j)}(k+1) \begin{bmatrix} \mathbf{0}_{M(j-1)\times M} \\ \mathbf{D}_{fq2}^{(j)}(k+1) \\ \mathbf{0}_{M(N-j)\times M} \\ \mathbf{E}_f^{(j)}(k+1) \end{bmatrix};$$

5. Obtaining $\mathbf{Q}_\theta^{(j)}(k+1)$ and $\gamma_j(k+1)$:

$$\begin{bmatrix} 1/\gamma_j(k+1) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta^{(j)}(k+1) \begin{bmatrix} 1/\gamma_{j-1}(k+1) \\ -\mathbf{a}^{(j-1)}(k+1) \end{bmatrix};$$

6. Joint estimation:

$$\begin{bmatrix} \mathbf{e}_{q1}^{(j)}(k+1) \\ \mathbf{d}_{q2}^{(j)}(k+1) \end{bmatrix} = \mathbf{Q}_\theta^{(j)}(k+1) \begin{bmatrix} \mathbf{e}_{q1}^{(j-1)}(k+1) \\ \lambda^{1/2}\mathbf{d}_{q2}^{(j)}(k) \end{bmatrix};$$

7. Obtaining the *a priori* error:

$e_j(k+1) = e_{q1}^{(j)}(k+1)/\gamma_j(k+1)$;

} % for $j$

} % for $k$

Finally, in order to adjust the equations of steps 1–3 of the algorithms in Tables 6.4 and 6.5 to this formulation, it suffices to observe that they can be split up into blocks that will be processed in an order-recursive way. The resulting lattice (or order-recursive) versions of the block-type MC-FQRD-RLS algorithms based on *a posteriori* and *a priori* backward prediction errors are summarized in Tables 6.6 and 6.7, respectively.

## 6.6 Application Example and Computational Complexity Issues

In this section, the effectiveness of the algorithms addressed in this work is illustrated in a non-linear filtering problem. In addition, we provide a brief discussion on computational complexity issues.

### *6.6.1 Application example*

The computer experiment consists of a non-linear system identification. The plant is a simple truncated second-order Volterra system [2] which can be summarized as

$$d(k) = \sum_{n_1=0}^{L-1} w_{n_1}(k)x(k-n_1) + \sum_{n_1=0}^{L-1}\sum_{n_2=0}^{L-1} w_{n_1,n_2}(k)x(k-n_1)x(k-n_2) + \rho(k). \quad (6.100)$$

Equation (6.100) can be easily reformulated as a multichannel problem with $M = L+1$ channels, where the most recent sample of the $i$th channel is

$$x_i(k) = \begin{cases} x(k), & i = 1, \\ x(k)x(k-i+2), & i = 2,\ldots,L+1, \end{cases}$$

and the $i$th channel order is

$$N_i = \begin{cases} L, & i = 1,2, \\ L-i+2, & i = 3,\ldots,L+1. \end{cases}$$

In the experiment, we have used $L = 4$ and the resulting multichannel system is depicted in Figure 6.5. The forgetting factor was set to $\lambda = 0.98$, and the power of the observation noise $\rho(k)$ was chosen such that the signal-to-noise-ratio (SNR) is 60 dB. The learning curves of the multichannel FQRD-RLS algorithms are compared to the normalized least-mean-squares[6] (NLMS) [15–17] and the result is plotted in Figure 6.6 for an average of 100 independent runs. The trade-

[6] The updating of the coefficient vector for the NLMS algorithm was performed according to

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \frac{\mu}{\sigma + \|\mathbf{x}(k)\|^2}\mathbf{x}(x)e^*(k),$$

where $\mathbf{x}(k)$ is the input signal vector, $\sigma$ is a small constant, and parameter $\mu$ was set equal to 1.

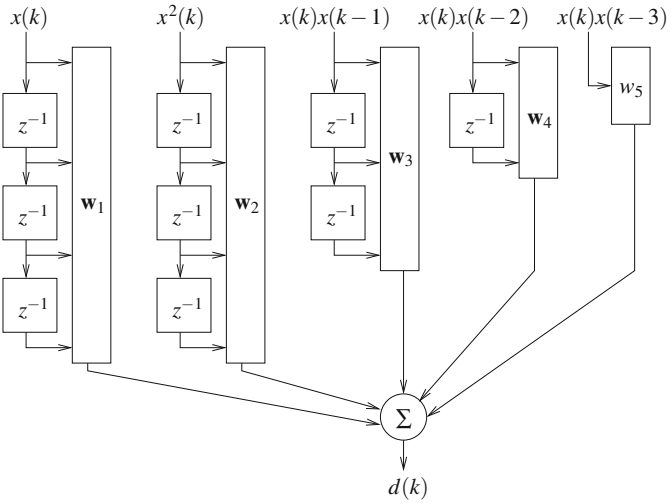


**Fig. 6.5** Multichannel set-up for a truncated second order Volterra system, $L = 4$.

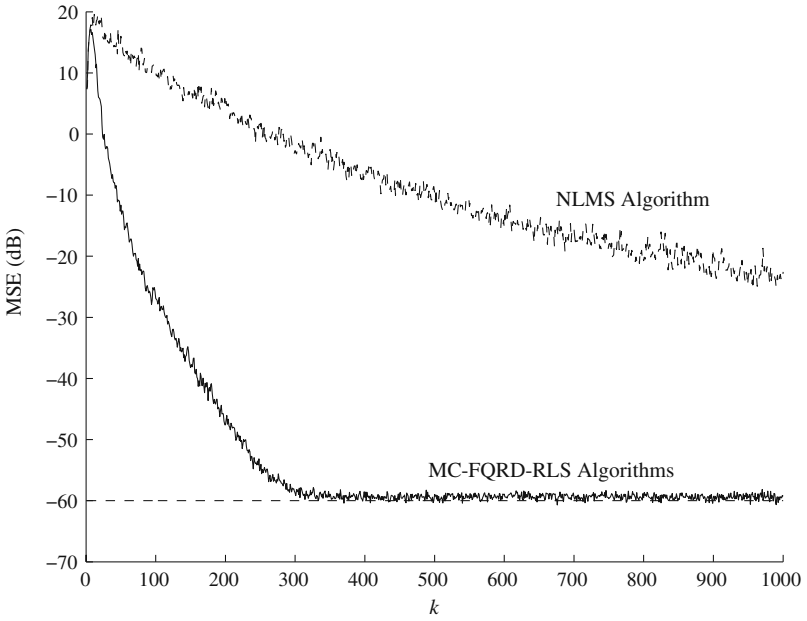


**Fig. 6.6** Learning curves of the non-linear system identification experiment.

off between computational complexity and speed of convergence is also illustrated in that figure.

### 6.6.2 Computational complexity issues

The computational complexity of the MC-FQRD-RLS algorithms, in terms of multiplication, divisions, and square roots per input sample, is summarized in Table 6.8, according to the classification introduced earlier in Table 6.1. Generally speaking, when the same structure and approach are considered, algorithms based on the *a posteriori* backward prediction errors updating have lower computational burden when compared to their *a priori* counterparts.

The suitability of the lattice structure for real-time implementations comes at the cost of a slight increase in the computational burden of the algorithms. On the other hand, sequential-type algorithms $\mathcal{O}[MP]$ computational complexity is lower by one order as compared to the $\mathcal{O}[M^2P]$ block-type multichannel algorithms computational complexity. It is also evident that the MC-FQRD-RLS algorithms outperform the conventional QRD-RLS and inverse QRD-RLS algorithms, $\mathcal{O}[P^2]$, in terms of computational costs, while maintaining the good numerical stability features. The computational advantage of block-type MC-FQRD-RLS algorithms is increasing for larger number of coefficients per channel, $N$, ($P = MN$ for channels of equal orders).

**Table 6.8** Computational complexity of MC-FQRD-RLS algorithms, according to Table 6.1.

| Algorithm | Multiplications | Divisions | Squared roots |
|---|---|---|---|
| Algs. **2, 4** [5–7], summarized in Table 6.4 | $4NM^2+11NM+5M^2+6M+7N-(4M^2+6M)\sum_{i=1}^{M}(p_i-i)$ | $2NM+2M+N-2M\sum_{i=1}^{M}(p_i-i)$ | $2NM+M+N-2M\sum_{i=1}^{M}(p_i-i)$ |
| Algs. **10, 12** [6, 7, 11], summarized in Table 6.5 | $4NM^2+11NM+5M^2+6M+9N-(4M^2+6M)\sum_{i=1}^{M}(p_i-i)$ | $2NM+3M+2N-2M\sum_{i=1}^{M}(p_i-i)+2$ | $2NM+M+N-2M\sum_{i=1}^{M}(p_i-i)$ |
| Alg. **1** [4], summarized in Table 6.6 | $4M^3N+17M^2N+12MN+5M^2+5M$ | $2M^2N+3MN+2M$ | $M^2N+2MN+M$ |
| Alg. **9** [11] summarized in Table 6.7 | $4M^3N+17M^2N+14MN+5M^2+6M$ | $2M^2N+5MN+3M$ | $M^2N+2MN+M$ |
| Algs. **6, 8** [10], summarized in Table 6.2 | $14NM+13M+5N-9\sum_{i=1}^{M}p_i$ | $3NM+4M-3\sum_{i=1}^{M}p_i$ | $2NM+3M-2\sum_{i=1}^{M}p_i$ |
| Algs. **14, 16** [11] summarized in Table 6.3 | $15NM+14M+5N-10\sum_{i=1}^{M}p_i$ | $4NM+5M-4\sum_{i=1}^{M}p_i$ | $2NM+3M-2\sum_{i=1}^{M}p_i$ |
| Algs. **5, 7** [9] | $14NM+13M+5N-9\sum_{i=1}^{M}p_i$ | $4NM+5M-4\sum_{i=1}^{M}p_i$ | $2NM+3M-2\sum_{i=1}^{M}p_i$ |
| Algs. **13, 15** [11] | $15NM+14M+5N-10\sum_{i=1}^{M}p_i$ | $5NM+6M-5\sum_{i=1}^{M}p_i$ | $2NM+3M-2\sum_{i=1}^{M}p_i$ |

## 6.7 Conclusion

The MC-FQRD-RLS algorithms exploit the time-shift structure in each channel to reduce the computational complexity of the conventional QRD-RLS algorithm which is of order $\mathcal{O}[P^2]$, $P$ being the number of coefficients. This chapter introduced various MC-FQRD-RLS algorithms based on the updating of the backward prediction error vector. Channels are allowed to have arbitrary orders, which enables us to deal with more general multichannel systems, e.g., Volterra systems. The algorithms presented in this chapter were derived using two distinct approaches: (1) *sequential approach* where channels are processed individually in a sequential manner, and (2) *block approach* that jointly processes all channels. Considering the case of $M$ channels, the computational complexities associated with block and sequential algorithms are $\mathcal{O}[MP]$ and $\mathcal{O}[M^2P]$, respectively. That is, taking a sequential approach will render the lowest complexity. The main advantages of the block algorithms are that they favor parallel processing implementations and can easily be turned into an order-recursive form. To clarify the differences among the many versions available in the literature, we provided a classification of these algorithms and their associated computational complexities.

**Acknowledgements** This work was partially funded by the Academy of Finland, Smart and Novel Radios (SMARAD) Center of Excellence and by the Buskerud University College (HIBU), Department of Technology (ATEK), Norway.

## References

1. N. Kalouptsidis and S. Theodoridis, Adaptive Systems Identification and Signal Processing Algorithms. Prentice-Hall, Upper Saddle River, NJ, USA (1993)
2. V. J. Mathews and G. L. Sicuranza, Polynomial Signal Processing. Wiley-Intercience: John Wiley & Sons, New York, NY, USA (2000)
3. A. L. L. Ramos, J. A. Apolinário Jr., and S. Werner, Multichannel fast QRD-LS adaptive filtering: Block-channel and sequential-channel algorithms based on updating backward prediction errors. Signal Processing, vol. 87, no. 7, pp. 1782–1798 (July 2007)
4. A. L. L. Ramos and J. A. Apolinário Jr., A lattice version of the multichannel FQRD algorithm based on *a posteriori* backward errors. 11th International Conference on Telecommunications, ICT'2004 (LNCS3124), Fortaleza, Brazil, vol. 1, pp. 488–497 (August 2004)
5. M. G. Bellanger and P. A. Regalia, The FLS-QR algorithm for adaptive filtering: the case of multichannel signals. Signal Processing (EURASIP), vol. 22, no. 2, pp. 115–126 (February 1991)
6. C. A. Medina S., J. A. Apolinário Jr., and M. G. Siqueira, A unified framework for multichannel fast QRD-LS adaptive filters based on backward prediction errors. 45th Midwest Symposium on Circuits and Systems, MWSCAS'2002, vol. 3, pp. 668–671, Tulsa, USA (August 2002)
7. A. L. L. Ramos, J. A. Apolinário Jr., and S. Werner, A general approach to the derivation of block multichannel fast QRD-RLS algorithms. European Signal Processing Conference, EUSIPCO'2005, Antalya, Turkey, vol. 1, pp. 1–4 (September 2005)

8. M. A. Syed and V. J. Mathews, QR-decomposition based algorithms for adaptive Volterra filtering. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 40, no. 6, pp. 372–382 (June 1993)
9. A. L. L. Ramos, J. A. Apolinário Jr., and M. G. Siqueira, A new order recursive multiple order multichannel fast QRD algorithm. 38th Midwest Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, USA, vol. 1, pp. 965–969 (November 2004)
10. A. L. L. Ramos and J. A. Apolinário Jr., A new multiple order multichannel fast QRD algorithm and its application to non-linear system identification. XXI Simpósio Brasileiro de Telecomunicações, SBT'2004, Belém, Brazil, vol. 1, pp. 1–4 (Setember 2004)
11. A. A. Rontogiannis and S. Theodoridis, Multichannel fast QRD-LS adaptive filtering: new technique and algorithms. IEEE Transactions on Signal Processing, vol. 46, no. 11, pp. 2862–2876 (November 1998)
12. M. A. Syed, QRD-based fast multichannel adaptive algorithms. International Conference on Acoustics, Speech, and Signal Processing, ICASSSP'91, Toronto, Canada, vol. 3, no. 6, pp. 1837–1840 (April 1991)
13. M. Harteneck, J. G. McWhirter, I. K. Proudler, and R. W. Stewart, Algorithmically engineered fast multichannel adaptive filter based QR-RLS. IEE Proceedings Vision, Image and Signal Processing, vol. 146, no. 1, pp. 7–13 (February 1999)
14. G. H. Golub and C. F. Van Loan, Matrix Computations. 3rd edition The Johns Hopkins University Press, Baltimore, MD, USA (1996)
15. P. S. R. Diniz, Adaptive Filtering: Algorithms and Practical Implementation. 3rd edition Springer, New York, NY, USA (2008)
16. S. Haykin, Adaptive Filter Theory. 4th Edition Prentice-Hall, Englewood-Cliffs, NJ, USA (2002)
17. A. H. Sayed, Fundamentals of Adaptive Filtering. John Wiley & Sons, Hoboken, NJ, USA (2003)